APPLE II 繪圖遊戲系列②

宗世麟 著

從APPLE II 到 圖形遊戲製作

GRAPHIC GAME

- 海空大戰
- 太空迷陣
- 打 磚 塊
- 繪 圖 板
- 魔術方塊



金字塔……

暢銷書「第三波」作者托佛勒教授以「第三波」稱呼資訊時代，並以「波前(Wave front)」代表鼓動第三波風潮一些前進的事物。

無疑的，在台灣電腦與管理事業書籍的出版是第三波新文明的一「波前」之一，她將負起引導台灣社會邁向第三波的重大責任。

然而坊間電腦書籍，內容不盡理想，浪費讀者寶貴的時間與金錢。本公司有鑑於此，乃邀請專家學者有系統的撰寫一系列電腦與管理書籍，以最高品質。服務讀者，同時達到「波前出版，必屬好書」的口碑。

波前電腦管理圖書有限公司

● 專訪蘋果BASIC編譯器(1)語言系統篇
楊國雄　編著　　定價 240 元
● 蘋果的數學國中上篇
楊國雄　編著　　定價 150 元
● 從TOOL KIT 看 APPLE系統
彭之瑞　著　　定價 210 元
● CP/M與PACKAGE大全
吳國寶　編著　　定價 220 元
● CP/M MBASIC 入門與實例
李華敏　彭淇湘　編著　定價 170 元
● APPLE Ⅱ高級BASIC 結構化設計
鐘健堯　著　　定價 155 元
● APPLE Ⅱ磁片COPY大全第 2 册
唐兆立　游允毅　編著　定價 160 元
● APPLE Ⅱ硬體應用專案設計
曾振賢　編著　　定價 160 元
● APPLE Ⅱ DOS 高級技巧
宗世麟等編著　　定價 215 元
● APPLE Ⅱ BASIC 繪圖
吳國寶　著　　定價 170 元
● CP/M作業系統入門
李華敏　著　　定價 180 元
● APPLE Ⅱ PASCAL 實用大全
吳國寶　編著　　定價 210 元
● APPLE Ⅱ BASIC 程式設計入門
胡文美　著　　定價 140 元
● APPLE Ⅱ 組合語言演習
章豪　編譯　　定價 130 元
● APPLE Ⅱ 機械人與無接點程式控制
吳占鰲著　　定價230元
● PFS 個人檔案系統與實例設計
環台電腦編譯　　定價 80 元
● BASIC 結構化程式設計
閔國田譯　　定價200元

● APPLE Ⅱ 組合語言
(1) LISA活用法
(增訂版・增加大量實例)
李天佑編著　　定價150元
(2) TOOL KIT活用法
徐寶龍編著　　定價 70 元
(3) CP/M 活用法
杜文宗等編著　　定價120元
● APPLE Ⅱ 世界程式競賽選輯
林和賢著　　定價150元
● 蘋果的智慧　黃松榮著　定價150元
● APPLE Ⅱ 硬體介面實驗初步
徐寶龍等編譯　　定價160元
● APPLE Ⅱ 組合語言精華
李天佑著　　定價230元
● 磁碟機・印表機活用法
胡文美著　　定價120元
● APPLE Ⅱ 軟硬體改良設計
彭之瑞等編著　　定價220元
● APPLE Ⅱ 硬體線路分析
李士彬等著　　定價 290元
● APPLE Ⅱ 程式師手册
宗世麟編譯　　定價190元
● APPLE Ⅱ 微電腦徹底研究系列
(一)入門篇　(二)進階篇　(三)磁碟系統篇
(五)動畫技巧篇　施威銘著定價220元
● APPLE Ⅱ 磁片COPY大全
梁振宇等編著　　定價160元
● APPLE Ⅱ 圖形遊戲系統設計
林和賢著　　定價180元
● 從APPLE Ⅱ 到圖形遊戲製作
宗世麟著　　定價180元
● APPLE Ⅱ 自動控制實務設計
吳占鰲編著　　定價250元

序　言

目前家用電腦風行台灣，許多家庭均擁有一部微電腦，其功能之繁多及精巧，更激起了許多人對微電腦的學習興趣。本書即根據這種需要，並以目前使用最廣的APPLE Ⅱ為主，希望能藉著大家所熟悉的娛樂遊戲程式，逐步探討，使讀者能了解APPLE Ⅱ微電腦的結構、遊戲程式的製作原理與程式設計的觀念及技巧。

由於本書的目的並不是一本遊戲程式集，故範圍從軟體的設計到硬體的結構均包含在內，並為了使讀者能徹底了解遊戲程式的製作方法吸收前人的長處及技巧，書中對每一部份均作了十分詳盡的說明。

全書主要分為三部份，第一部份對所有必具的知識均加以討論，甚至對於APPLESOFT亦以系統的眼光來說明，使讀者不僅會使用BASIC語言，更能了解 BASIC語言的內部結構。第二部份則討論一些在製作遊戲程式時，常用的一些工具(即程式)，如音樂的產生、及高、低解像圖形的製作程式。以幫助讀者，使能輕易的製作出屬於自己的遊戲程式。第三部份，搜集了10個遊戲程式，涵蓋了大部份 BASIC 遊戲程式的種類，許多都是市面上包含在錄音帶或磁碟片上的遊戲程式，書中排列乃按難易程度，由淺入深排列。

因為本書所列出的各類遊戲程式並非很容易，因此作者對各

程式均說明的十分詳細，使只要略通 BASIC 程式設計的人均能輕易的看懂這些程式。

最後，作者要感謝鐘健堯同學提供資料及鼎力相助使本書能如期出版。

宗世麟

于 1983 年 4 月 20 日

目　　錄

第一部份　基本概念

第一章　APPLE Ⅱ之一般概念 ………………………… 1

第一節　APPLE Ⅱ記憶區的配置 1

第二節　螢幕與第一頁文字幕記憶區 7

第二章　了解APPLESOFT ………………………… 19

第一節　記憶區之使用 19

第二節　BASIC命令句的結構 28

第三節　APPLESOFT的變數 32

第四節　位址重整與其他 39

第三章　低解像圖面 ………………………… 43

第一節　畫面結構 43

第二節　監督程式中有關低解像圖形的副程式 45

第四章　高解像圖面 ………………………… 51

第一節　高解像螢幕與記憶區的對應關係 52

第二節　高解像畫面結構 56

第三節　圖形的製作 63

第四節　圖形表的使用 71

第五節　APPLESOFT中的幾個高解像繪圖副程式 79

第二部份　幾個應用上的觀念

第一章　APPLE 音效的產生 ………………………… 85

第二章　如何分割 APPLESOFT BASIC 程式 ………… 93

第三章　低解像圖形表的設計及製作 ……………… 105

第四章　高解像圖形表的製作 ……………………… 115

第三部份　遊戲程式

程式一　APPLE 音樂（APPLE MUSIC）…………… 127

程式二　雷射槍（LAZER BLASTER）……………… 135

程式三　APPLE COLOR DEMONSTRATION ………… 151

程式四　太空迷陣（SPACE MAZE）………………… 163

程式五　繪圖板程式（APPLE PAINTBOX）………… 179

程式六　海空大戰（AIRSEA BATTLE）…………… 201

程式七　打磚塊（LITTLE BRICK OUT）…………… 229

程式八　魔術方塊（MAD, MAD CUBE）…………… 257

程式九　金字塔（PYRAMID）……………………… 289

程式十　五子棋（GOBANG）………………………… 321

附錄一：

INTEGER BASIC 和 APPLESOFT 之比較 ………… 341

附錄二：

參考書目 …………………………………………… 351

第一部份

基本概念

第一章　APPLE Ⅱ 之一般概念

第二章　了解APPLESOFT

第三章　低解像圖面

第四章　高解像圖面

第一章　APPLE Ⅱ之一般概念

隨著資訊工業的進步，電腦已成爲人類生活中不可或缺的要件，尤其目前個人用電腦（PERSONAL　COMPUTER）已深入每個家庭中。最初它的功用也許只是娛樂或是一些簡單的應用，但經過一段時間後，必然會興起自己編寫程式的興趣，至此使用者可能會發現，各家用電腦廠商所提供的程式語言均與其機器本身的系統及硬體結構有密切關聯。因此若想在程式編寫上發揮機器的最大能力，則首先必須對所使用的系統內部結構有深刻的認識才行。

基於此點，本書首先在本章中，對目前使用最普遍的APPLE Ⅱ電腦的內部結構做一番描述。但因本書是針對APPLE的遊戲程式（game)，故所介紹者均爲與此應用性程式有相關性的結構，至於其他較低階的硬體結構則不在本書討論範圍，由於本書所討論的game 程式並非簡單的應用性程式，相信本章所討論的結構已足夠應付讀者的需要。

第一節　APPLE Ⅱ記憶區的配置

APPLE Ⅱ 以6502 微處理機做爲中央處理單元，故其輸入／輸出爲memory-map Ⅰ／0 的型式。即所有輸入／輸出埠被視爲記憶區，其所佔的位址也屬於記憶區的一部份，故可將其記

憶體分爲三大部份：讀／寫記憶區（RAM），僅讀記憶區（ROM），及輸入／出埠（I/0 port）。此外APPLE將256個位元組劃分爲一頁（page），所以完整的64K記憶體共可分爲256頁，並按次序分別賦予0到255的頁數編號。由第0頁至第191頁共48K位元組（10^4＝1024位元組稱爲1K位元組），屬於讀／寫記憶區；第192頁至207頁共4K位元組（ROM）的位置，則供輸入／出埠及各週邊裝置使用，最後的12K位元組，自第208頁至第255頁爲僅讀記憶區，儲存一些常用的系統程式，提供使用者使用但不能更改其內容。在APPLE Ⅱ PLUS 這部份包含了一APPLESOFT（BASIC 解譯程式）及系統監督程式（MONITOR）。（請參照圖1-1）

系統記憶體結構圖		
頁 10進位	數 16進位	
0	$00	RAM（48K）
1	$01	
2	$02	
⋮	⋮	
⋮	⋮	
190	$BE	
191	$BF	
192	$C0	I/O（2K）
193	$C1	
⋮	⋮	
⋮	⋮	
198	$C6	
199	$C7	
200	$C8	I/O ROM（2K）
201	$C9	
⋮	⋮	
⋮	⋮	

206	$CE	
207	$CF	
208	$D0	ROM（12K）
209	$D1	
⋮	⋮	
⋮	⋮	
254	$FE	
255	$FF	

圖 1-1　記憶體配置圖

由於讀／寫記憶區不但可以讀出資料又可存入資料，故為使用者存放程式的區域，而僅讀記憶區，僅能讀出系統預先存好的程式，故其用途便有了限制。但在APPLE Ⅱ中，使用者可藉著插入一片語言卡（ language card ），（具有額外的 16 k RAM ）可取代原有的僅讀記憶區，使讀／寫記憶區擴展為 64 k 位元組，供其他方面的使用。如我們想執行PASCAL 所編寫的程式，則記憶區中除了要有使用者所編寫的PASCAL 語言應用程式外，還必須存入PASCAL的編譯程式（ compiler，用來將PA-SCAL語言轉換成機器所能執行的較低階語言），此時原來僅讀記憶區所存之BASIC解譯程式，對我們而言便毫無用處，便可利用此語言卡，取代此僅讀記憶區。

雖然APPLE Ⅱ 具有 48 k 位元組的讀／寫記憶區，但使用者也不是可全部隨意使用。第 0 頁（ $0000 － $00FF ）在AP-PLE Ⅱ 中為十分特殊的記憶位置。在 64 k 記憶體的機器，若想利用與記憶位址有關的組合語言指令（ Assembly instruction ）表示出某一位址，須使用 16 位元（ $2^{16} = 64 \times 1024 = 64$ k ）即兩個位元組；但對APPLE Ⅱ 而言，若想表出第 0 頁的位址，只須使用 8 個位元（ $2^8 = 256 = 1$ page ）便可（詳細情形，請參

考書後所附組合語言指令表所列出的各指令所佔的位元組數），故監督程式及BASIC解譯器均用此頁儲存其所使用到的資料，以減少所佔的記憶空間及加快執行速度。

第一頁（＄0100－＄01FF）的256個位址爲系統堆疊區（stack），做爲6502一些特殊指令所使用的暫存區。如PHA，PHP PLA、PLP均爲單一位元組指令，僅表出動作（operation）而沒有指出動作執行（operand）之位址，即代表其動作執行位址爲系統已知，無須再指明。而這些"已知位置"便是在此堆疊區的特定位置；此外在程式中常用到的CALL、GOSUB及RETURN指令亦利用到此頁記憶區。如在一BASIC程式中：

```
        :
 20   GOSUB 100
        :
100   GOSUB 500
110   X = X + 1
        :
500     :        ┐
        :        ├ 一副程式
700   RETURN     ┘
        :
```

GOSUB 500後，程式便跳至500去執行，最後到700的RETURN時，系統是如何知道應回去執行110的x＝x＋1而不是到20行的下一個指令呢？這便是因系統在由100跳到500去執行時，便已先將此GOSUB 500的下一個指令的起始位址存入此堆疊區中，RETURN時只須再將此位址取出（POP）便可按正常的次序執行。諸如此類的例子還有許多，因不在此書討論範圍，故不再詳述。

第二頁（＄0200－＄02FF）爲鍵盤輸入緩衝區，將由鍵盤送過來的ASC Ⅱ電碼存入此緩衝區中，直到“RETURN”爲止；例如我們在鍵盤上鍵入＊400.500 RETURN時，系統會由4、0、……逐字“收集”，而第二頁就用來暫時存放這些“收集”的字元，直到使用者鍵入“RETURN”通知系統命令已輸入完畢後，系統才加以處理，並準備再從＄0200起收集下一道命令。但因一頁僅具有256個位址，故每一行程式連RETURN在內，不能超過256個字數（空格也算）。因此若將此鍵盤輸入緩衝區擴展成爲512個位址，則每一行程式的字數最多便可達到512個字了，但因實際上很少有這麼長的命令，且每行程式太長了便不易閱讀，故通常均將此鍵盤輸入緩衝區定爲256個位址。

第三頁（＄0300－＄03FF）的256個位址中，監督程式僅使用最後的16個位元組，故大部份的位址均可由使用者自由使用。通常game程式便常利用此部份儲存一些不太長的機器語言程式，讀者可在往後的程式中發現這一點。

第四頁至第七頁（＄0400－＄07FF）共1K位元組的位址，爲供字幕或低解像圖面第1頁所使用；而在第8頁至第11頁（＄800－＄BFF）的1K位元組位址，則供字幕或低解像圖面的第2頁所使用，但通常第8頁爲使用者使用區域的起始頁數，即我們很少利用此區域顯示低解像圖形或文字幕。

雖然自＄800起至記憶體結束位址，均可由使用者存入程式或資料，但若程式中使用到高解像畫面，則記憶體自＄2000至＄3FFF共8K位址便須由高解像畫面的第一頁所使用，而＄4000到＄5FFF共8K位址便由高解像畫面的第2頁所使用。綜合以上所述可歸納如表1-1。

表 1-1　APPLE Ⅱ記憶區之配置

<table>
<tr><th rowspan="2">頁數
(DEC)</th><th colspan="2">位址</th><th colspan="2" rowspan="2">用途</th></tr>
<tr><th>10 進位</th><th>16 進位</th></tr>
<tr><td>0</td><td>0－255</td><td>$0－$0FF</td><td colspan="2">系統程式使用（MONITOR，BASIC…）</td></tr>
<tr><td>1</td><td>256－511</td><td>$100－$1FF</td><td colspan="2">系統堆疊區</td></tr>
<tr><td>2</td><td>512－767</td><td>$200－$2FF</td><td colspan="2">鍵盤輸入緩衝區</td></tr>
<tr><td>3</td><td>768－1023</td><td>$300－$3FF</td><td colspan="2">自由使用及監督程式使用</td></tr>
<tr><td>4
5
6
7</td><td>1024－2047</td><td>$400－$7FF</td><td colspan="2">文字幕及低解像圖面第 1 頁</td></tr>
<tr><td>8
9
10
11</td><td>2048－3071</td><td>$800－$BFF</td><td>文字幕及低解像圖面第 2 頁</td><td rowspan="6">自由使用區</td></tr>
<tr><td>12
至
31</td><td>3072－8191</td><td>$C00－$1FFF</td><td></td></tr>
<tr><td>32
至
63</td><td>8192－16383</td><td>$2000－$3FFF</td><td>高解像畫面第 1 頁</td></tr>
<tr><td>64
至
95</td><td>16384－24575</td><td>$4000－$5FFF</td><td>高解像畫面第 2 頁</td></tr>
<tr><td>96
至
191</td><td>24576－49151</td><td>$6000－$BFFF</td><td></td></tr>
<tr><td rowspan="2">192</td><td>49152－49279</td><td>$C000－$C07F</td><td colspan="2">特殊內部 I/O 位址</td></tr>
<tr><td>49280－49407</td><td>$C080－$C0FF</td><td colspan="2">週邊插座使用</td></tr>
<tr><td>193 至 199</td><td>49408－51199</td><td>$C100－$C7FF</td><td colspan="2">週邊 ROM</td></tr>
<tr><td>200 至 207</td><td>51200－53247</td><td>$C800－$CFFF</td><td colspan="2">週邊擴展用 ROM</td></tr>
<tr><td>208 至 247</td><td>53248－63487</td><td>$D000－$F7FF</td><td colspan="2">BASIC 解譯程式 ROM</td></tr>
<tr><td>248 至 255</td><td>63488－65535</td><td>$F800－$FFFF</td><td colspan="2">監督程式 ROM</td></tr>
</table>

第二節　螢幕與第一頁文字幕記憶區

一部電腦可能不具有畫面結構，但文字幕卻是各電腦所必須的，用來與使用者相互溝通。因此在APPLE Ⅱ開機之後，監督程式會立刻設定螢幕爲文字幕形式，並輸出一閃爍的空白字形在螢幕上，告訴使用者下一個輸入文、數字的顯現位置，此即一般所稱的游標（cursor）。故當使用者輸入命令至電腦時，實際上電腦要做下列幾件事：首先測試鍵盤是否有輸入？若有則依所輸入的字鍵，將其存入文字幕中適當位置（或採取適當動作），而顯示在螢幕上，並令游標向右移動一個位置，當然還要將此文、數字或特殊符號以ASCⅡ的電碼形式存入第2頁（$200－$2FF），以逐字搜集命令，供系統執行。

可能說到這裡有些讀者還不了解文字幕是什麼？簡單的說文字幕，就是位於系統讀／寫記憶區的一塊位址，電腦硬體能自動將其內容顯示在螢光幕上，故我們按其意義稱它爲"幕"。由於具有此文字幕儲存了使用者的輸入字鍵，所以電腦能很簡單的在此位址上從事一些修改先前輸入字鍵或跳行的工作（如當使用者按下"→"，"←"或RETURN等鍵時）。但在正式討論文字幕的記憶區結構時，我們想先對APPLE Ⅱ 的輸入／出字形編碼做一番說明。

事實上；一部電腦內部所能儲存的只是以0,1構成的一些二進位型式的數字資料，因此爲了要能將文字幕記憶區的資料以字形、符號的方式印在螢幕上，便須對這些字形、符號定出一套對應的編碼，如此才能使電腦硬體在讀到193時能在螢幕上顯現"A"，讀到163時顯示出"＃"。目前較爲大家所熟知的是一套ASCⅡ編碼，但APPLE Ⅱ所使用的編碼方式與ASCⅡ 編碼略

有不同。在監督程式中，APPLE 是以負的ASCⅡ碼（即ASCⅡ碼加上128）方式存入。這種方式使得儲存此輸入碼的記憶位址的最高位元（bit 7），若有輸入則變爲1，否則便爲0。當然APPLE的監督程式在取出此輸入碼後，必須將其最高位元再變爲0。控制這些動作主要有二個位址：一爲$C000（−16384）即爲APPLE 的鍵盤按鍵字碼的存入位址（keyboard data input)，若其內含值超過127（即bit 7爲1）則代表有鍵被按了；當然將此值減去128即是標準的ASCⅡ 值。此外還有一個位址爲$C010（−16368）爲鍵盤旗標（clear keyboard strobe），用來將鍵盤觸發（strobe）訊號歸零。即當鍵盤上有鍵被按了，−16384 位址上的第7位元被設定爲1，必須再經由LDA $C010的指命（或POKE−16368，0），將其第7位元歸零，以便繼續讀入其他按鍵。因爲APPLE 的監督程式，在測試是否有按鍵輸入時，僅測試$C000位址的第7位元是否爲1，而不管其內容。故使用者應特別注意，若程式中由−16384（$C000）位址讀取資料後，應記得將鍵盤觸發訊號歸零（POKE−16368，0或LDA $C010)，才能接受到下次新的按鍵輸入字碼。

我們知道APPLE 每個位址可存8個位元資料，2^8爲256故每個位址所能表示的碼數共有256 種，但螢幕上可印出的字形只有64個（見表1-2）。所以APPLE 將低於128的碼分成兩部份，分別對應於反白字形及閃爍字形各64 個；而將128至159的碼編當做特殊控制碼；160～223的碼供64個正常字形之用；及224至255的碼保留給小寫字母使用（見表1-2）。但因APPLE Ⅱ的字形產生器無法產生小寫字母，故一般的APPLE Ⅱ的這些小寫字母只是重覆64個正常字形而已。此外讀者

應注意表1-2是螢幕字形碼，在由鍵盤輸入的字碼中沒有反白及閃爍的字形，只有正常字形，故由鍵盤輸入時，其輸入鍵碼必大於 127 ，不會有小於 128 的情形。且讀者應清楚一項事實，並不是每一個螢幕字形碼皆能顯現在螢幕上，如CONTROL -G僅能使喇叭發聲，而不會在螢幕上產生動作。

表 1 - 2

	Inverse				Flashing				Normal							
									(Control)						(Lowercase)	
Decimal	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
Hex	$00	$10	$20	$30	$40	$50	$60	$70	$80	$90	$A0	$B0	$C0	$D0	$E0	$F0
0 $0	@	P		0	@	P		0	@	P		0	@	P		0
1 $1	A	Q	!	1	A	Q	!	1	A	Q	!	1	A	Q	!	1
2 $2	B	R	"	2	B	R	"	2	B	R	"	2	B	R	"	2
3 $3	C	S	#	3	C	S	#	3	C	S	#	3	C	S	#	3
4 $4	D	T	$	4	D	T	$	4	D	T	$	4	D	T	$	4
5 $5	E	U	%	5	E	U	%	5	E	U	%	5	E	U	%	5
6 $6	F	V	&	6	F	V	&	6	F	V	&	6	F	V	&	6
7 $7	G	W	'	7	G	W	'	7	G	W	'	7	G	W	'	7
8 $8	H	X	(	8	H	X	(	8	H	X	(	8	H	X	(	8
9 $9	I	Y	)	9	I	Y	)	9	I	Y	)	9	I	Y	)	9
10 $A	J	Z	*	:	J	Z	*	:	J	Z	*	:	J	Z	*	:
11 $B	K	[	+	;	K	[	+	;	K	[	+	;	K	[	+	;
12 $C	L	\	,	<	L	\	,	<	L	\	,	<	L	\	,	<
13 $D	M	]	–	=	M	]	–	=	M	]	–	=	M	]	–	=
14 $E	N	^	.	>	N	^	.	>	N	^	.	>	N	^	.	>
15 $F	O	_	/	?	O	_	/	?	O	_	/	?	O	_	/	?

討論了APPLE的輸入及螢幕字形碼後，我們便可接著研究文字螢幕與記憶區的對應關係。在上一節中我們提過APPLE II的文字幕與低解像圖面共同使用＄400至＄7FF共1 K 的位置。而文字幕與低解像圖面結構在與記憶區的對應關係上大致是相同的；且在編寫game 程式中有時無法很容易的用PLOT 等一些繪圖指令將複雜的圖形畫出或控制圖形的移動，此時若能明白螢幕與記憶位址的對應關係，便可很容易的解決此問題。

在文字幕中，APPLE II 能顯示24行文字且每行能印出

40個字形，即總共能印出 24 × 40 = 960 個字。而每個字用去一個儲存位址，共用了960個位址，但文字幕的記憶區共佔1024個位置，還有 1024 − 960 = 64 個位址呢？我們發現它是以8個位址爲一組共分8組，每組位址提供給一個週邊插座（sot)使用，此外螢幕上的次序是否正好與記憶區的順序相配合呢？事實上不是。我們可做個小小的試驗，讀者可試試下列指令：

```
 0  CALL  -936
10  FOR  I=0  TO  255
20  POKE  1024+I , I
30  NEXT  I
40  VTAB  20 : END
```

在上面程式中，將螢幕字形表中的各字形按文字幕記憶區的位址順序分別存入，以便由執行中發現一些有趣的現象。我們可發現：第一、螢幕上24列文字並非依位址順序排列，而是將其分爲三部份（1～8列、9～16列、17～24列）。顯現時每一部份依次序輪流各印一行後再重覆，即第1列印過後，印第9列、後再印第17列，然後再印第2列……。第二、有一部份字形沒有顯示在螢幕上（如8、9、：、；、＝、？）。因此儲存這些字形碼的位址可能在960 個顯現位址之外，即此960個顯現位址並非連續，而是將64個週邊插座所用的位址分散在其間。事實上文字幕的位址結構如表1-3。由表1-3(a)我們看到當第 17 列顯示完後，以下 120 ～127等 8 個字形碼（8、9、:、;、<、＝、>、？）被存入＄478至＄47F，而此位址爲供I/O 之用，而不是用來做螢幕顯示之用的，這解釋了爲什麼有些字形會不見了的原因。

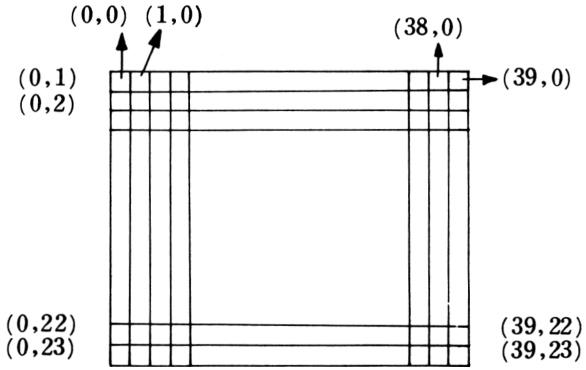
表 1-3　文字幕位址分配

Table No. 1 - In Memory Order Sequence			Table No 2 - In Screen Line Sequence		
DEC:	HEX:	LINE No.	DEC:	HEX:	LINE No.
1024-1063	400-427	1	1024-1063	400-427	1
1064-1103	428-44F	9	1152-1191	480-4A7	2
1104-1143	450-477	17	1280-1319	500-527	3
1144-1151	478-47F	*	1408-1447	580-5A7	4
1152-1191	480-4A7	2	1536-1575	600-627	5
1192-1231	4A8-4CF	10	1664-1703	680-6A7	6
1232-1271	4D0-4F7	18	1792-1831	700-727	7
1272-1279	4F8-4FF	*	1920-1959	780-7A7	8
1280-1319	500-527	3	1064-1103	428-44F	9
1320-1359	528-54F	11	1192-1231	4A8-4CF	10
1360-1399	550-577	19	1320-1359	528-54F	11
1400-1407	578-57F	*	1448-1487	5A8-5CF	12
1408-1447	580-5A7	4	1576-1615	628-64F	13
1448-1487	5A8-5CF	12	1704-1743	6A8-6CF	14
1488-1527	5D0-5F7	20	1832-1871	728-74F	15
1528-1534	5F8-5FF	*	1960-1999	7A8-7CF	16
1536-1575	600-627	5	1104-1143	450-477	17
1576-1615	628-64F	13	1232-1271	4D0-4F7	18
1616-1655	650-677	21	1360-1399	550-577	19
1656-1663	678-67F	*	1488-1527	5D0-5F7	20
1664-1703	680-6A7	6	1616-1655	650-677	21
1704-1743	6A8-6CF	14	1744-1783	6D0-6F7	22
1744-1783	6D0-6F7	22	1872-1911	750-777	23
1784-1791	6F8-6FF	*	2000-2039	7D0-7F7	24
1792-1831	700-727	7	1144-1151	478-47F	*
1832-1871	728-74F	15	1272-1279	4F8-4FF	*
1872-1911	750-777	23	1400-1407	578-57F	*
1912-1919	778-77F	*	1528-1535	5F8-5FF	*
1920-1959	780-7A7	8	1656-1663	678-67F	*
1960-1999	7A8-7CF	16	1784-1791	6F8-6FF	*
2000-2039	7D0-7F7	24	1912-1919	778-77F	*
2040-2047	7F8-7FF	*	2040-2047	7F8-7FF	*

(a)　　　(b)

明白了文字幕與記憶位址的對應關係後，我們知道並不是我們直覺上所認為的 1024 個位址中的前 960 個位址對應到螢幕上

的1－24列（如第2列即由1024＋（2－1）＊40 位址開始），而比我們想像的要複雜。因此我們必須找出一個方法能很容易的算出螢幕上某一位置會對應到記憶區的那一個位址。首先我們按照APPLE Ⅱ的螢幕型式定出一個座標系，為方便計算，則將



X座標自0至39對應於每列上的40個位置，Y座標自0至23對應於螢幕上的第1列至24列。座標系決定後，我們不難得到一個公式：

對應位址＝1024＋（Y MOD 8）＊128＋INT（Y／8）＊40＋X。

因文字幕記憶區由＄0400（1024）開始，很顯然的座標（X，Y）＝（0，0）正對應到位址1024。又前面說過文字幕按記憶區位址順序分為3部份（0～7列，8～1列，16～23列 —按新座標），將每一部份的第一列（即第0、8、16列）印完後才開始印每一部份的第二列（即第1、9、17列）……以此類推，又由表1-3中我們可知印完每一部份的第一列，須佔去128 個位址，即每一部份中每列間的位址差為128，故由（Y MOD 8），可求出Y座標位在其所在部份的第幾列（0～7），而將（ Y

MOD 8）＊128＋1024 便可得出開始印第一部份的（Y MOD 8）列時的起始位址；再由（Y／8）—其值爲 0 至 2—可求出此第 Y 列位於第幾部份（由 0 ～2 等三部份）。而第一部份的第 n 列與第二部份的第 n 列位址僅相差 40，第二部份之第 n 列與第三部份之 n 列亦僅相差 40（n＝0…7），故將第一部份之第（Y MOD 8）列之起始位址加上（Y／8）＊40 便可得出座標 Y 列所在部份的第（Y MOD 8）列的起始位址。最後再加上 X 座標（因同一列中的 40 個螢幕位置在記憶區中爲依序排列）便可導出上式。例如我們想求出（X，Y）＝（18，5）的對應位置，首先由（18 MOD 8）＝2 知道第 18 列位在某部份之第 2 列，且 1024＋2＊128＝1280 得出第一部份第 2 列的起始位址爲 1280，又由 Y／8＝2 可知第 18 列是在第三部份，而第三部份的第 2 列與第一部份的第二列位址相差 2＊40，故知第 3 部份的第 2 列（即第 18 列）的起始位址爲 1280＋2＊40＝1360；且 X＝5 爲 1360 算起第 6 個位置（因 X 由 0 算起），故知（X，Y）的對應位址爲 1365。

上面講了這麼多，讓我們看看下面的程式，讀者不妨將其鍵入 APPLE II 中，看是否能達到我們所想要的結果與上面的理論做個印證。

```
10  REM  SPECIFY THE X,Y COORDINATE OF
         THE SCREEN AREA
20  INC=0
100 GOTO 130
110 IF HT<40 THEN RETURN
120 HT=1: VT=VT+1: IF VT>18 THEN END
```

```
125  RETURN
130  CALL-936 : REM HOME
140  POKE 34,18 : REM SET WINDOW
150  CALL-936
160  INPUT "INPUT CHAR" ; CHT$
170  CHT=ASC(CHT$)
180  IF INC>0 THEN 200
190  INPUT "INPUT VT,HT";VT,HT
200  GOSUB 110
210  SCRN=1024+(VT-INT(VT/8)*8)*128 +
         INT(VT/8)*40+HT
220  POKE SCRN,CHT :PRINT CHT :
     INC=INC+1 : HT=HT+1 : GOTO 160
```

以上程式可將使用者指定的字形，在指定的螢幕位置顯示出來。程式中的VT代表Y座標（自0到23），HT 代表X座標（自0到39）。在此程式中只接受一次使用者的指定位置，以後便將使用者指定的字形，按螢幕次序各別印出。此外讀者可能注意到，在螢幕上出現的好像都是反白和閃爍字形，這是怎麼一回事呢？這是因爲在程式 170 CHT＝ASC（CHT$），用了函數ASC的緣故，讀者可參考APPLESOFT BASIC PROGRAMMING REFERENCE MANUAL 的第60 頁找到滿意的解釋。

以上討論的好像都是在BASIC程式方面的應用，那麼在組合語言（或機器語言）中是否也適用呢？當然，只要機器的結構確定了，各種應用總是相似的，只是若使用組合語言，情形便簡

單多了。在組合語言的程式中我們不必自己算出螢幕座標在記憶區所對應的位址，而可將這件工作交給監督程式代勞。監督程式中有二個副程式COUT及VTAB分別在＄FDED與＄FC22的位址，並利用＄24（CH）及＄25（CV）的內容，決定所印文字在螢光幕上的位址。那CH及CV是做什麼用呢？簡單的說：CH的內容代表所欲印出文字與設定之螢幕窗（scroll window）的水平距離，而CV代表所欲印出文字與螢幕頂端（screen top）的垂直距離，兩者的作用與BASIC中的HTAB及VTAB相同（讀者可再參閱本節最後之討論）。VTAB副程式可計算出螢幕位址在文字幕記憶區的對應位址，而COUT可將在A暫存器中所存的文字顯現在螢幕上的指定位置。讓我們試試下列程式：

```
10  HOME
20  POKE 36,10 : REM CH
30  POKE 37,15 : REM CV
40  POKE 0,193 : REM 193 stands for "A"
50  CALL 768   : REM $300
60  END
```

在程式中，我們將所要顯示的文字“A”存入＄0，並將CH及CV設定為10與15（第15列第10行），最後呼叫＄300中的副程式執行。當然在執行前必須先將下列機器語言存入＄300:

```
* 300 L
0300-   20 22 FC    JSR  $FC22
0303-   A5 00       LDA  $00
0305-   20 ED FD    JSR  $FDED
0308-   60          RTS
```

在呼叫COUT（$FDED）前，必須先將$00所存的文字放入A暫存器中，因COUT並不會自動至$00取值，只是直接將A暫存器中的內容顯示在螢幕上。

由上面的例子，我們可發覺監督程式中有許多副程式可提供使用者使用，只要我們了解它們並善加利用，必可達到事半功倍的效果，這一點讀者可在往後的例子中發現。

最後在結束本節之前，再將與第一頁文字幕（或螢幕）有關係的位址列舉出來，供讀者參考：

32 $20 WNDLFT：

為文字窗（scroll window）的最左邊位置，範圍由0至39（$27）。

33 $21 WNDWDTH：

為文字窗的寬度，範圍由1至40－（WNDLET）。

34 $22 WNDTOP：

為文字窗的頂端位置，對全文字幕（full text screen）時其範圍由0至22（$16），對文字／圖形混合幕（mixed GR／TEXT）時，範圍由22至22。當執行HOME 指令後，系統會將游標置於此位置所指列數之最左端。

35 $23 WNDBTM：

通常為文字窗底線的下一個位置，範圍由（WNDTOP）+1至24（$18）

以上位置在開機時，監督程式自動設定為0，40，0，24。故若使用者的程式重新設定這幾個位址的內容，但在reset後，經過監督程式的warm start 過程，又將被存入原來的系統設定值0，40，0，24。

我們看看下面例子

```
10 POKE 33,5
20 FOR I=1 to 10
30 POKE 32,I
40 PRINT "1234567"
50 NEXT I
```

讀者可發現由於POKE 33,5 的作用，使文字幕上只印出了“12345”。再RESET並鍵入] PRINT“ABCDE”，發現文字窗又被系統重新設定(“ABCDE”出現在第0行)

36 $24 CH:

其內容代表所欲印出文字與 WNDLFT 之水平距離，範圍自0至(WNDLFT)-1。故其內容與設定的文字窗相關，但:

37 $25 CV:

其內容爲所欲印出文字在螢幕(非文字窗)上的列數(即垂直距離)，其範圍自0至23($17)。

此外監督程式中有許多可供使用者直接使用的副程式，讀者可參考APPLESOFT-BASIC REFERENCE MANUAL，本書則在使用到時加以說明。

第二章　了解APPLESOFT

要運用一種語言，首先必須要知道語言的語法才能寫出一份完整無誤的程式；但若想收事半功倍的效果，則應再“了解”這種語言。有鑑於此，本書將爲讀者介紹一些 APPLESOFT 的基本概念及一些APPLESOFT的技巧。詳細閱讀本章不但能幫助我們了解game 程式，若能善加利用，更能使讀者寫出有“深度”的程式。當然，作者更希望能藉著本章，提升讀者對程式語言的學習範圍，使讀者能看的愈遠，想得愈深。但本章只是APPLESOFT 的一部份，有興趣的讀者可參考書後所列參考書籍之Ref 5，以求更深一層的了解。

第一節　記憶區之使用

APPLESOFT 對記憶體的使用方式，如圖 2-1，圖右邊的一些變數，是APPLESOFT所使用的指標。APPLESOFT的指標通常位於 0 頁區（＄00 ～＄FF ），由指標的內含值我們可知道各部份記憶區的邊界，及一些很有用的資料（讀者稍後可發現）。改變這些指標的內含值，或將指標內容交換，常可幫助我們進行偵錯或由讀者自行設定記憶區的使用方式（這一點極爲重要）。

圖 2-1 爲 48k 記憶體機器的使用情形，使用者的APPL-

ESOFT BASIC 程式是由＄800（2048）向上存，故圖中由＄800 開始。

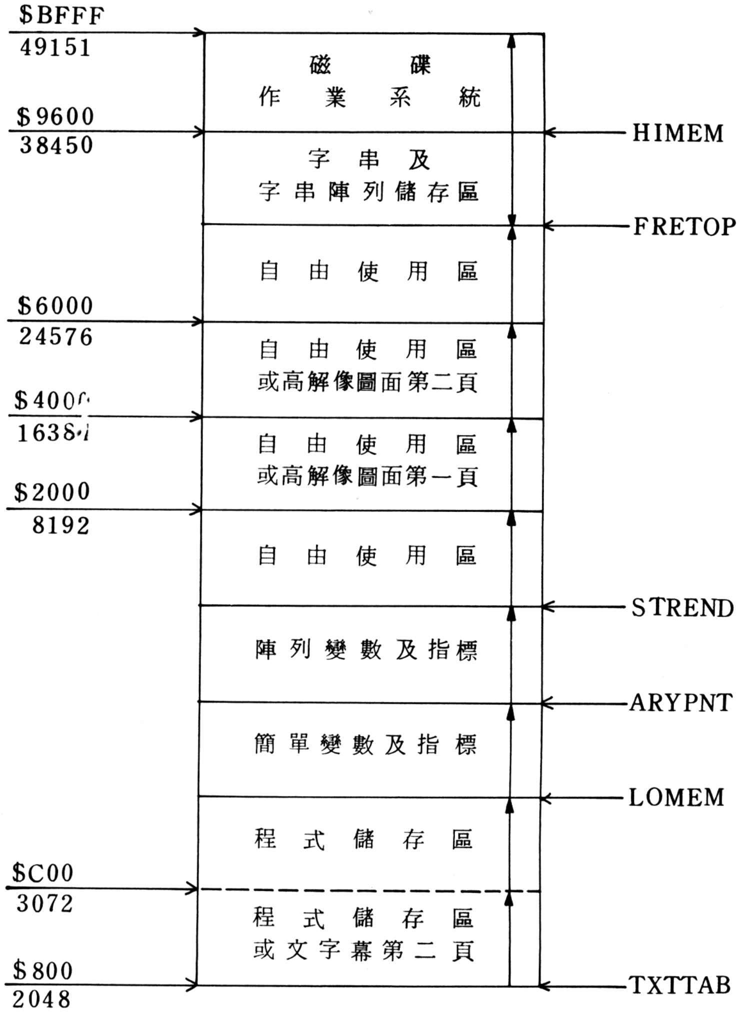


圖 2-1 APPLESOFT之記憶體配置圖

首先介紹APPLESOFT 的一些零頁指標，除了圖上所表示出來的外，並加入一些稍後將會用到的指標：

TXTTAB($67,68/103,104)：

其內含值爲使用者BASIC程式的開始位址，APPLESOFT 解譯器將其自定爲$801(2049)。我們可移動此指標指向較高位址，以空出一塊"安全"區域供存放機器語言程式或圖形表(shape table)。

VARTAB($69,6A/105,106)：

即圖中之LOMEM，爲簡單變數、字串指標及函數指標的開始儲存位址。可藉著LOMEM:的命令改變其內容。

ARYPNT($6B,6C/107,108)：

爲陣列變數及字串陣列指標的開始儲存位置。

STREND($6D,6E/109,110)：

已使用記憶區之最後一個位址，亦即自由使用區的開始位置。(見圖2-1)

FRETOP($6F,70/111,112)：

自由使用區的結束位址，亦即字串及字串陣列儲存區的結束位址(因字串由HIMEM 向下存放)。(見圖2-1)

MEMSIZ($73,74/115,116)：

即圖中之HIMEM，爲APPLESOFT 所能使用的記憶體最高位址加1，亦即字串及串列的起始儲存位置(加1)，然後向下往低位址儲存。

CURLIN($75,76/117,118)：

爲正在執行的命令句之行數號碼。

OLDLIN($77,78/119,120)：

前一個已執行命令句的行數號碼。

OLDTXT($79,7A/121，122):

指著前一個已執行之命令句的最後一個位元組的位址(此位址之內含值爲0)——後面將會說明。

DATLIN($7B,7C/123,124):

其內含值爲提供BASIC中READ命令句所正在輸入資料的DATA命令句之行數。

DATPTR($70 7E/125，126):

指著下一個將被使用的資料(在DATA命令句中)之起始位址。

INPPTR($7F,80/127,128):

若在INPUT命令句時，則指向鍵盤輸入緩衝區之$201位址；若在READ命令句時，則指向當時所使用的DATA命令句。

VARNAM($81,82/129,130):

存放著目前所使用到的變數名稱。

VARPNT($83,84/131,132):

指著目前所使用的變數之位址或在字串變數的情形中，指向說明此字串長度的位元組。

PRGEND($AF,BO/175，176):

指著使用者程式的最底端後二個位址(見後面說明)。這個指標在合併兩個程式成爲一個程式時特別有用。

下面的程式可將APPLESOFT幾個指標之內含值顯現在螢幕上:

```
100  REM LOMEM: AND PROGRAM
         POINTER WILL READ THE
         SAME UNLESS LOMEM: HAS
         BEEN CHANGED
```

```
900  HOME : DIM A$(115): GOSUB 10
     50
1000  FOR I = 103 TO 115 STEP 2:A
      = A + 1: PRINT I;" ";I + 1;
     " = "; PEEK (I) +  PEEK (I +
     1) * 256;A$(I): NEXT : PRINT
     "175 176 = "; PEEK (175) +  PEEK
     (176) * 256;A$(8)
1025  STOP
1050 A$(103) = "  START OF PROGRA
     M":A$(105) = "  LOMEM: PTR":
     A$(107) = "  VARIABLE PTR":A
     $(109) = "  CURRENT MEMORY":
     A$(111) = " END OF STRINGS":
     A$(113) = " GENERAL PURPOSE
     PTR":A$(115) = " APPLESOFT H
     IMEM":A$(8) = "  PROGRAM POI
     NTER"
1100  RETURN
```

其中HIMEM（即MEMSIZ）的顯示值在沒有磁碟作業系統的情況下將爲49152。

以下，我們再看看二個常可利用到的指標（0頁位址）及副程式。（除了本節所列出的幾個指標及副程式外，還有一些可常利用到的指標及副程式，我們留待相關章節中再加以討論）。

TXTPTR（$B8，B9／184，185）：

在執行程式時，APPLESOFT必須逐一讀取字元及指令代碼（見後），便是利用此指標指出所將要讀取的字元碼及指令代碼的位址。

FORPNT（$85，86／133，134）：

爲一般用途的指位器，APPLESOFT的COPY 副程式使用到此位址，通常我們可視程式的需要做一般性的使用。在下面程式中是用來存放第一個變數的位址。

JUMPSTAR（$D43C）：

讀者可能已發現在監督程式控制下若鍵入“0G”則控制權

轉回APPLESOFT，這便是因為在＄00-02中存有JUMP ＄D43C；而＄D43C便為APPLESOFT的一進入點。

CHRGET（＄00B1）：

此副程式可由程式中找出下一個字元或指令碼。前面說過TXTPTR指著下一個將要讀取的字元碼或指令碼的位址，若利用JSR ＄00B1，可將下一個字元或指令碼讀入A暫存器中，且若此字元為一終止符號（LF或：）則設定"Z"旗標（Zero-flag）；若為非數字性字元則設定借位旗標（Carry Flag）。

SNERR（＄DEC9）：

印出"SYNTAX ERROR"並停止程式之執行。

PTRGET（＄DFE3）：

為尋找變數位址的副程式，執行時會將TXTPTR所指的變數之值存放的位址，依高低位元組的次序存於Y及A暫存器及VARPNT指標內。

CHKCLS（＄DEB8）：

檢查目前TXTPTR所指的字元是否為"）"。

CHKOPN（＄DEBB）：

檢查目前TXTPTR所指的字元是否為"（"。

CHKCOM（＄DEBE）：

檢查目前TXTPTR所指的字元是否為"，"。

SYNCHR（＄DEC0）：

檢查目前TXTPTR所指的字元是否與A暫存器中的字元相同。以上4個副程式將會不斷的將TXTPTR向前移動，直到碰到所要的符號為止。

下面我們介紹一個程式，此程式可用以交換兩字串變數的內含值：

```
]LIST

100  REM      "STRING SWAP"
          BY RANDY WIGGINGTON

110 HEX$ = "3B0: 20 E3 DF 85 85 8
     4 86 20 BE DE 20 E3 DF A0 02
      B1 85 48 B1 83 91 85 68 91
     83 88 10 F3 60 N D823G": REM
     GET ASBLY LANG INTO KBD BUFF
     ER

120  FOR I = 1 TO  LEN (HEX$): POKE
     511 + I, ASC ( MID$ (HEX$,I,
     1)) + 128: NEXT : POKE 72,0:
      CALL  - 144: REM DUMP BUFF
     TO MEMORY

130  POKE 1013,76: POKE 1014,176:
      POKE 1015,3: REM SET & VECT
     OR
140  REM   ROUTINE IS COMPLETELY
     RELOCATABLE WITH NO ABSOLUTE
     REFS EXCEPT TO APPLESOFT ROM
     IT CAN RUN JUST AS WELL AT
     $800, $2000,  OR ???

200  REM   FORMAT TO USE IT AS
           FOLLOWS
210  REM  ..:&STRING1$,STRING2$:.

220  REM EXAMPLES

230  REM &A$,B$
240  REM  $A$(J+9,K/4),B$(I)

300  REM HERE IS ANOTHER EXAMPLE

310  INPUT "FIRST STRING>";A$
```

```
320  INPUT "SECOND STRING>";B$
330  PRINT A$,B$
340  & A$,B$
350  FOR I = 1 TO 500: NEXT
360  GOTO 330: REM

370  REM THIS SWAP USES NO ADDIT-
     IONAL STRING SPACE, THUS
     AVOIDING GARBAGE COLLECTION
```

程式中第340行對使用者所輸入的字串變數$A，$B 進行交換。當APPLESOFT碰到“ & ”時會將TXTPTR指在$ A上，並自動JUMP至$ 3F5；而在程式中第130行已事先將 JUMP $ 3B0 的指令POKE至$ 3F0中，故會繼續執行$ 3B0 中的機器語言（此機器語言在程式一開頭的第110，120行已先POKE至位址$ 3B0開始的記憶區中）。那位於$ 3B0的機器語言究竟做些什麼呢？首先將其換成組合語言：

```
03B0   20 E3 DF   JSR   $DFE3
03B3   85 85      STA   $85
03B5   84 86      STY   $86
03B7   20 BE DE   JSR   $DEBE
03BA   20 E3 DF   JSR   $DFE3
03BD   A0 02      LDY   #$02
03BF   B1 85      LDA   ($85),Y
03C1   48         PHA
03C2   B1 83      LDA   ($83),Y
03C4   91 85      STA   ($85),Y
```

```
03C6    68        PLA
03C7    91 83     STA    ($83),Y
03C9    88        DEY
03CA    10 F3     BPL    $03BF
03CC    60        RTS
```

在JSR $DFE3的指令中，APPLESOFT 將存變數$A之字串長度的位址依高低次序存入Y，A 暫存器中。而在此程式中我們再將其存入($85，$86)一般用途指位器中，由JSR$DEBE 越過BASIC程式第340行中的“，”，再JSR $DFE3，APPLESOFT 會將存字串$B之字串長度的位址存入VARPTR及Y，A暫存器中(此時Y，A暫存器中所存的值已沒有必要保留)；以下的指令便是將$A及$B 的字串長度及眞正存此字串的位址(共占三個位元組)互相交換，便可使$A，$B字串內容互換。

看到這裡如果還不太了解，沒關係！請繼續向下看，在後面我們會介紹幾種簡單變數的結構幫助您了解！

第二節　BASIC 命令句的結構

APPLESOFT解譯器本身只是用組合語言所編寫的一份非常巨大的解譯程式，而主要的一項工作便是解譯位於＄801使用者所輸入的BASIC程式。

若我們鍵入：］100　PRINT　A，則APPLESOFT解譯器會將其改爲如圖2-2的方式儲存。

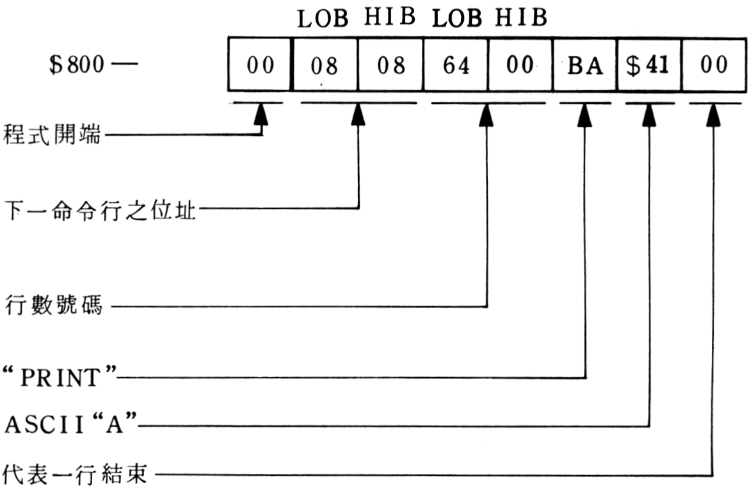


圖 2-2　命令句之結構

圖2-2中，"PRINT"用＄BA代替，因在APPLESOFT中所使用的爲標準ASCII碼（值均小於128），並將大於及等於128的107個數碼用來表示各指令代碼，而"PRINT"的指令代碼便是＄BA（其他各指令的指令代碼讀者可參閱參考書籍

Ref 2之第121頁附錄F）。

由圖2-2中我們可發現在APPLESOFT除了表示出 100 PRINT A幾個字之外，還有幾個多餘的位元組。事實上APPLESOFT對程式中每一行（同屬於一行數號碼之下的所有敘述句），除了表示出指令代碼、變數名稱或ASCⅡ字元的各位元組外，還多用了5個位元組，儲存APPLESOFT所必須知道的資料。

由圖2-2可知，頭兩個位元組表示出下一個命令行的起始位址。當一開始進入APPLESOFT後，APPLESOFT便將這兩個位元組設定為0，故若接受到LIST命令時，APPLESOFT便檢查這兩個位元組，若其值為0，便知記憶體中沒有程式存在；相反的，若發現其值非0，則知至少有一行程式存在而將此行顯示出來，往後便按此位址繼續搜尋下一行，直到兩個位元組均為0時便知道已到程式的盡頭而停止。第3、4 個位元組用來表示行數號碼，除了LIST命令用到之外，如DATA、GOTO 、GOSUB等命令均須用到。接著這四個位元組後均為有關程式之資料，若其值小於128均代表ASCⅡ碼，大於127則為BASIC中的鍵字（指令代碼）——在APPLESOFT所有非關鍵字（non-Keyword）的字元均以標準ASCⅡ表示。最後一個位元組，其值為$00，用以表示出一行的結束。最後我們還須注意到，圖2-2 的第一個0（位址$800）並不屬於BASIC程式，而是一個前導位元組，由APPLESOFT中的LOAD 命令所使用。

以下為讀者介紹一個程式EXAM。利用此程式，讀者只要輸入想檢查的BASIC程式中的某一行數號碼，此程式便會將此行以標準APPLESOFT的儲存型式（如圖2-2）顯示出來，並輸出 16 進位的存放位址，可供讀者在監督程式之下修改APPL-

ESOFT BASIC的程式內容（如修改行數號碼）。

```
LIST

10  REM

* EXAM    BY VAL J GOLDING *

CALL -A.P.P.L.E. IN DEPTH 1

62000  DEF  FN M(X) = X -  INT (X
        / 256) * 256:S$ = "01234567
       89ABCDEF":ZERO$ = "0"
62010  INPUT "EXAM ";L0: IF L0 <
        = 0 THEN  END
62020  GOSUB 63998:LST = LST + 4:
       EXAM$ =  STR$ (L0)
62030  IF  LEN (EXAM$) <  > 5 THEN
       EXAM$ = ZERO$ + EXAM$: GOTO
       62030
62040  FOR I = 1 TO 5: POKE LST +
       I, ASC ( MID$ (EXAM$,I,1))::
        NEXT : GOSUB 63999
62050  GOSUB 62110: IF LINE > L0 THEN
       62090
62060 : PRINT " ";:K% = LOC / 256
      : GOSUB 62100:K% =  FN M(LOC
      ): GOSUB 62100: PRINT ": ";
62070  FOR I = 1 TO 8:K% =  PEEK
      (LOC): GOSUB 62100: PRINT "
      ";:LOC = LOC + 1: IF LOC = (
      NXTLIN) THEN  PRINT : GOTO 6
      2010
62080  NEXT : PRINT : GOTO 62060
62090  PRINT "* NO LINE ";L0;" *"
      : GOTO 62010

62100 J% = K% / 16:K% = K% - 16 *
      J% + 1:J% = J% + 1: PRINT  MID$
      (S$,J%,1); MID$ (S$,K%,1);::
       RETURN
62110 LOC =  PEEK (103) +  PEEK (
      104) * 256
62120  GOSUB 62160
62130  IF NXTLIN = 0 THEN LINE =
      65536: RETURN
62140  IF LINE >  = L0 THEN  RETURN

62150 LOC = NXTLIN: GOTO 62120
62160 NXTLIN =  PEEK (LOC) +  PEEK
      (LOC + 1) * 256:LINE =  PEEK
      (LOC + 2) +  PEEK (LOC + 3) *
      256: RETURN
63998 LST =  PEEK (121) +  PEEK (
      122) * 256:LST =  PEEK (LST +
      1) +  PEEK (LST + 2) * 256::
       RETURN
```

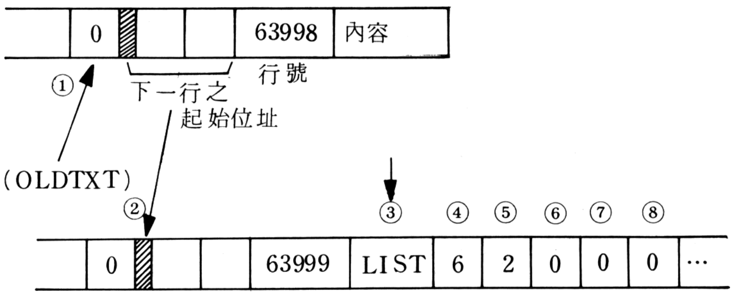
```
63999  LIST 62000: RETURN

]RUN
EXAM 62150

62150 LOC = NXTLIN: GOTO 62120

 0A5D: 73 0A C6 F2 4C 4E 43 D0
 0A65: 4E 58 54 4C 49 4E 3A AB
 0A6D: 36 32 31 32 30 00
EXAM
```

程式中第 62010 行的 LO，用以儲存所欲檢視的行數號碼，第 62020 中 GOSUB 至 63998 利用 OLDTXT（121，122）指位器找出開始儲存第 63988 行的位址並存於變數 LST 中，再利用 PEEK（LST+1）+PEEK（LST+2）* 256 找出 63999 行開始的位址。我們可用圖形說明如下：



LST=PEEK（121）+PEEK（122）* 256
使 LST 指在①的位址
故 LST=PEEK（LST+1）+PEEK（LST+2）* 256
便可將 LST+1，LST+2 的內容存入 LST 使
LST 再指到②的位址

由此可知63998行與63999 行必須相鄰（這就是爲什麼兩者行號僅相差1），且LIST必須爲 63999 行的第一個指令，該行的行號也必須是5位數。

在62020行中，又將LST＋4使LST指到圖中③的位址。然後將變數LO利用STR$ 換成字串形式存入EXAM$ 變數中，並檢查長度是否爲5，不是則在字串前補上“0”。在62040行中我們將字串EXAM$中的5個數值字元逐字存入圖中④，⑤，⑥，⑦，⑧的位置中，如此便更改了63999 行中原來命令句的內容，達到我們的目的。

62050 行GOSUB至 62110 行，利用TXTTAB（103，104）指位器，從程式在記憶體的開端檢查行數號碼是否爲使用者所指定的，不是則再利用每行中所存的下一行位址，逐行搜尋直到找到爲止。

在62060行中，GOSUB至62100 行做16進位轉換後，以16進位方式印出。在將指定的行數印出後，又回到62010行接受新的行號，若使用者所輸入的行號小於或等於零則使程式終止。

第三節 APPLESOFT的變數

在第一節中提過VARTAB（即LOMEM：）指位器所指示的是變數開始儲存位址，但變數究竟是怎麼儲存的呢？APPL-ESOFT 是如何判別變數的形式呢？

APPLESOFT 主要有7種形式的變數，分別存於三個不同的記憶區中（見圖2-1）。又可將這7種型式歸納爲兩大類型：簡單變數與陣列變數（ARRAY），屬於簡單變數的有4種：實

數、整數、字串與函數（function）。要APPLESOFT 解譯器區別這些變數的類型，則在表示這些變數時必須給予不同的特徵。我們知道整數變數後加上“%”符號，字串變數後加上“$”符號均可用以區別，但在APPLESOFT解譯過程中，看到的並不是這些符號而是在表示變數名稱時所用的正值或負值的組合。

在BASIC程式中，我們最多可用到238個字元去組成一個變數名稱，但不論用多長的名稱，APPLESOFT只認識頭二個字元，如“LEADPIPE”和“LEAPYEAR”兩個變數名稱，APPLESOFT 會當成相同的變數“LE”；而對於僅由1個字元構成的變數名稱，APPLESOFT會在其後加上一個虛字元（即$00或$80視當時應用正值或負值而定）。

在第一節中我們提過一對指位器VARNAM（129，130）與VARPNT（131，132），我們常可利用這一對指位器來找出某一變數的相關資料。假設我們想找出某一程式中APPLESOFT存放變數B之值的位址，通常有二種方式：

```
10100   B = B : POKE 0, PEEK(131): POKE 1,
        PEEK(132): AD=PEEK(0)+PEEK(1)*
        256
```

與
```
10200   B = B : AD=(PEEK(131)+PEEK(132)*
```
256）−7 在第二個方法中，執行到AD=（PEEK(131)+PEEK(132)*256）−7 時VARPNT 指的位址已不是存變數B的位址，而是指在存變數AD的位址，故須減去7個位元組（?見後）重新指回變數B；而第一個方法已事先將存變數B之值的位址存入$0，$1中故不須減7個位元組。

下面讓我們正式開始說明幾種變數的儲存方式。

1. 實變數

圖 2-4 表示的是實變數的儲存方式。在APPLESOFT中對所有簡單變數均給予 7 個位元組的儲存位置，事實上僅有實變數將 7 個位元組全部用到，但爲求簡便，對簡單變數一律使用 7 個位元組的儲存位置。

頭二個位元組用的是正（標準）ASCⅡ碼，表示出實變數的名稱，第三個位元組爲指數（2的次方）部份值，接下去的四個位元組用來表示出假數部份（MANTISSA），由最高位向下存入。事實上，實數部份的表示法相當複雜，這裡僅略述其儲存方法，有興趣的讀者可自行參閱參考書目Ref 5。

EX：A=40000

位元組	說　明		16進位	10 進位
0	變數名稱之第一個字元（正ASCⅡ）		41	65
1	變數名稱之第二個字元（正ASCⅡ）		00	0
2	指　數		90	144
3	假　數 (mantissa)	(M.S.Byte) 最高位數	1C	28
4			40	64
5			00	0
6		L.S.Byte) 最低位數	00	0

圖 2-4 實變數

2. 整變數

整變數在APPLESOFT中，是將其變數名稱以負ASCⅡ

碼表示，而與其他變數區別，見圖2-5。在執行速度上整變數較實變數慢很多，因APPLESOFT在運算時須先將其化爲浮點表示法再進行運算，且在存放程式的記憶區中，整變數須多用一位元組表示"%"，故應有限度的使用。整變數其內含値的儲存方式爲先存高位元組，後存低位元組。

EX：B%=32767

位元組	說　　明	16進位	10進位
0	整變數名稱第一字元（負ASC Ⅱ）	C2	194
1	整變數名稱第二字元（負ASC Ⅱ）	80	128
2	高位元組之值	7F	127
3	低位元組之值	FF	255
4	未　使　用	00	0
5		00	0
6		00	0

圖2-5　整變數

3. 字串變數

APPLESOFT對字串變數將其名稱的第一字元以正ASCⅡ碼表示，而將其名稱的第二字元以負ASC Ⅱ表示。字串變數的內含值並不直接存於此變數表中，而在表中的第4、5位元組存入儲存其內容的起始位址，如圖2-6所示。第3位元組存放其內含值（字串）的長度。

事實上字串變數又可分兩種；第一種即程式中已設定其內容

的字串如：100 A$＝"HELLO"；第二種即其內容須藉INPUT 自鍵盤或磁碟中讀入者如：110 INPUT A$。兩者的差別在於儲存其內含字串的位置不同，對於第二種情形其內含字串由HIMEM：向下（低位區）儲存，這就是爲什麼要用指標（第4、5位元組）指著其開始儲存位址。

EX：CD$＝"HELLO"

位元組	說明	16 進位	10 進位
0	變數名稱之第一字元（正ASC II）	43	67
1	變數名稱之第二字元（負ASC II）	C4	196
2	內含字串之長度	05	5
3	字串儲存位址（高位元組）	FB	251
4	字串儲存位址（低位元組）	8F	143
5	未使用	00	0
6		00	0

圖 2-6　字串指標

4.函數變數（FUNCTION）

當函數在程式中定義時如DEF FN A（B）＝，APPLESOFT 會在變數表中建立佔7個位元組的變數指標，如圖2-7。且若變數名稱"B"事先未在程式出現過，APPLESOFT 也會在變數表中將"B" 定義爲實變數。若變數B事先已有定義則執行此函數後，並不會改變B之原值。

相同的，頭二個位元組儲存變數名稱，但第一字元以負ASC

Ⅱ碼表示，第二字元則以正ASCⅡ碼表示。第3、4位元組則指著程式記憶區中緊接著“＝”後的第一個位元組（即函數的定義）。第5、6位元組則指著儲存參數值的位址（在此例中則指著儲存變數B之值的位址）。第7位元組儲存“＝”後第一個字元的正ASCⅡ碼，並沒有什麼作用。

EX：DEF　FNEF（GH）＝XY

位元組	說　　明	16進位	10進位
0	變數名稱之第一字元（負ASCⅡ）	C5	197
1	變數名稱之第二字元（正ASCⅡ）	46	70
2	函數定義的位址（低位元組）	0E	14
3	函數定義的位址（高位元組）	08	8
4	〔GH〕儲存位址的低位元組	1B	27
5	〔GH〕儲存位址的高位元組	08	8
6	“＝”後第一個字元的正ASCⅡ碼	58	88

圖 2-7　函數指標

以上所介紹的爲簡單變數，往下再爲讀者介紹陣列變數的基本概念。

ARYTAB（＄6B，6C）及STREND（＄6D，6E）兩指位器所包含的記憶區爲陣列變數儲存區，見圖2-1。通常陣列分三類：實數、整數與字串陣列，三種仍與簡單變數的表法類似，APPLESOFT仍以正負ASCⅡ碼的組合來區別其種類。

圖2-7中爲一整數陣列變數，讀者可由此推知實數、字串陣列的情形。頭二個位元組表示陣列變數的名稱。第3、4位元組代表存放下一個陣列變數的位址與此陣列變數儲存位址之差距（如此陣列變數存於＄0900開始的位址，而其第3、4位元組內容爲＄21,＄00，則下一個陣列變數的儲存位置即爲由＄0921開始），接下來的第五個位元組指出此陣列爲若干維（DIMENSION）之陣列，若有N維，則接下去便會有n組（二個位元組爲一組），分別說明第N，N－1，N－2，……1維的大小，緊接著下去的才是陣列變數中每一元素之內含值。

EX：DIM JK％（2，3）

位元組	說　　明	16進位	10進位
0	變數名稱之第一字元（負ASCII）	CA	202
1	變數名稱之第二字元（負ASCII）	CB	203
2	至下一個陣列變數之位址差（低位元組）	21	33
3	至下一個陣列變數之位址差（高位元組）	00	0
4	維　　數	02	2
5	第2維之大小（高位元組）	00	0
6	第2維之大小（低位元組）	04	4
7	第1維之大小（高位元組）	00	0
8	第1維之大小（低位元組）	03	3

圖2-7　陣列變數

在圖2-7中，因陣列變數 JK%的12個單元值尚未設定故沒有列出，但已保留24個位元組用來儲存這12 個值（因圖2-7中第3、4位元組已指出下一個變數的儲存位址為33 個位元組之後，9＋24＝33）。

第四節　位址重整與其他

在前面曾提過字串變數的內含值（即字串）為由HIMEM向下存放，而由FRETOP（$6F，6A）指位器指著所有字串儲存區之結束位址。因在APPLESOFT BASIC 程式中，字串長度不須預先指明，加上各字串長度均不相同，使得 APPLESOFT無法預留字串儲存位址，故採用這種方式。

在圖2-8中，說明字串儲存方式。圖中第一次A$＝“HELLO”，已失去作用，由“TEST”所代替（APPLESOFT會將A$ 變數中之指標改變），但仍佔一部份的記憶位址。因此若BASIC程式中大量使用字串，最後會使FRETOP指位器與STREND 指位器重疊，而無法再存入字串，使得程式停止。在這種情形中，雖然字串儲存區中存滿了字串，但事實上有許多儲存區被一些已經失去作用的字串所佔滿，形成浪費（如圖2-8中所存的“HELLO”），為了容納更多新的字串，免得程式中止執行，APPLESOFT中有一個副程式叫“GARBAGE”，能進行位址重整。程式首先找出在最高位址的有效字串（如圖2-8中之B$），將其移至HIMEM：，並將其變數表中所指的位址改成移動後新的位址，依次再找出下一個佔次高位址的有效字串、進行同樣工作，如此進行下去便能將所有已失去作用的字串除去，使得所存的均為有效字串，並空出一些字串儲存位置後，再進

行使用者的BASIC程式。故若讀者在執行大量字串處理的程式時，發現程式執行中忽然暫停幾秒鐘，可不必吃驚，只是APPLESOFT 在進行位址重整。

```
EX: INPUT A$,B$,A$
    ? HELLO
    ? ?BYE
    ? ?TEST
```

	位址 HEX	位址 DEC	內含值 HEX	內含值 DEC	內含值 字元
舊A$	8FFF	36863	4F	79	O
	8FFE	36862	4C	76	L
	8FFD	36861	4C	76	L
	8FFC	36860	45	69	E
	8FFB	36859	48	72	H
B$	8FFA	36858	45	69	E
	8FF9	36857	59	89	Y
	8FF8	36856	42	66	B
A$	8FF7	36855	54	84	T
	8FF6	36854	53	83	S
	8FF5	36853	45	69	E
	8FF4	36852	54	84	T

圖 2-8 字串的儲存

最後我們介紹一些如何加快APPLESOFT BASIC 執行速度的基本觀念做爲本章的結束。

前面提過將程式註解行REM 與可執行的命令行合併爲一行可省掉每行中一些指標，節省記憶空間及加快速度。順序執行的

程式（即沒有GOTO），速度也較快但通常會使用較多的記憶空間。此外使用變數也可節省許多的執行時間，加快速度，且使用次數最多的變數，應儘量在程式的一開始便定義（關於這點在GAME程式中更應特別注意）。整變數的運算較實變數慢，因在運算時整變數須先化成實數再做運算；但在陣列的情形中，整變數陣列使用較少的記憶空間。

若在程式中，某一複雜的運算式經常會被用到，則爲求加快速度，則應儘可能將一些常數運算先求出，避免重覆做不必要的計算。事實上一些編譯程式（如PASCAL COMPiler）均有optimizer 程式，在執行前將這些工作做好，再轉成機器語言，如此便不會減低程式執行的速度，但由於APPLESOFT在根本是屬於解譯程式（interpreter），故無法做這些工作。此外APPLESOFT 允許迴圈後的NEXT 後不接變數名稱，但讀者應特別注意，這種方式常使得程式的執行速度減低約20％ 。

下面我們研究一個有關執行速度的例子。下列程式若存在程式的一開頭，則執行時間約需8秒；但若前面加入25行的REM註解行，則執行時間要約12秒。

```
 25   REM

100   X=X+1
110   IFX<1000 THEN 120
115   END
120   GOTO 100
```

下一個程式中，我們將TXTTAB（103，104）指位器改指到OLDTXT（121，122）所指的位址，使APPLESOFT以爲在此之前沒有程式存在，而達到加快執行速度的目的。

```
  0  LSOM=PEEK(103):HSOM=PEEK(104)
     REM THIS DATA SHOULD BE STORED BEFORE
           LINE 90 IS EXECUTE
                 :
                 : (ANY NUMBER OF PROGRAM LINES)
 90  AD=PEEK(121)+256*PEEK(122)+1:
     POKE 103,AD-INT(AD/256)*256:
     POKE 104,INT(AD/256)
100  X=X+1
110  IF X<1000 THEN 120
115  POKE 103,LSOM:POKE 104,HSOM:END
120  GOTO 100
```

以上程式，不論第90 行之前有多少行，執行時間大約爲8秒。第90行的主要作用爲利用OLDTXT 指位器，決定其在記憶區的位址，並 POKE 至TXTTAB中。

第三章　低解像圖面

第三章　低解像圖面

APPLE Ⅱ 提供兩種繪圖能力：高解像圖形及低解像圖形。一般遊戲程式均喜歡使用生動精彩的高解析圖形，而使大家忽略了低解析度圖形。事實上低解析度圖面製作容易，且色彩變化較多，在有些場合還是相當有用。相信讀者一定看過APPLE 的 COLOR DEMONSTRATION 程式中的萬花筒（KALEIDOSCOPE）非常精彩，令人眼花撩亂，而其圖形即爲低解像圖面；又如一些遊戲程式在開始前的標題畫面，亦多以低解像圖面製成。

第一節　畫面結構

第一章中提過低解像圖形第一頁的記憶區與文字幕記憶區共同使用＄400－＄7FF 的位址，故一旦將TEXT改設定爲低解像圖面後，影幕上便會出現一個個的彩色方塊，但若使用 GR 命令設定，會立刻將這些彩色方塊清除，故每次由TEXT利用 GR 轉換成低解像圖面，總有一瞬間的彩色雜亂圖形，才成爲全黑底色的螢幕；相同的再由 GR 利由“TEXT”命令回到文字幕上，則顯現的便是一些沒有意義的ASCⅡ 字元。由此可知文字幕與低解像幕對於相同的資料有不同的解釋方式。事實上在低解像記憶區中，每一位元組代表二個上下相疊的方塊圖形，即文字幕中的

每一個ASCⅡ字元代表圖形幕中的兩個相疊彩色方塊。因此，原來文字幕包括24行，每行40個字元的螢幕，在圖形幕時便成爲48行，每行40個方塊圖形的螢幕了。

由上面的說明，我們知道低解像記憶區中的每一位元組代表兩個圖形方塊，每一圖形方塊便由4個位元控制。由此可推知每一圖形方塊至多只能用16種顏色顯現出來。這16種顏色爲：

0	$0	黑色	8	$8	棕色
1	$1	深紅色	9	$9	橙色
2	$2	深藍色	10	$A	灰2色
3	$3	紫色	11	$B	粉紅色
4	$4	深綠色	12	$C	淺綠色
5	$5	灰1色	13	$D	黃色
6	$6	藍色	14	$E	水色
7	$7	淺藍色	15	$F	白色

但讀者應注意，在BASIC中我們用0－15的値來設定顏色，均能很正確的繪出。但在組合語言中便須特別注意了，由於低解像圖形記憶區中每一位元組代表二個相疊的方塊，0-3位元控制上方塊（即偶數行），4-7位元控制下方塊（奇數行），若我們指定顏色爲$01（即將$01存入$30的位址中——稍後說明）打算畫座標（0，0）的點，則不會有問題，能在（0，0）出現深紅色的方塊，但若打算畫在座標（1，0）的點，出現的便是黑色而不是深紅色了。因此若想利用組合語言畫低解像圖形，在設定顏色時最保險的方法便是將顏色設定爲$00，$11，$22，……$EE，$FF，則不論畫在奇數行或偶數行便都不會發生問題了。

既然文字幕與低解像圖形均使用相同的記憶區，則在低解像

圖形幕中螢幕與記憶區的對應方式也和文字幕時的情形相同，在此不再說明。

在使用BASIC程式繪低解像圖形時，我們不必考慮低解像圖形記憶區與螢幕的對應關係，只需指出螢幕上的位置座標即可。但在組合語言中，是否也如此的方便呢？是的，因爲在監督程式中有許多副程式可幫助我們做這些螢幕位置與記憶區位置的轉換，或畫圖的工作，使我們即使利用組合語言，仍能很輕易的繪出低解像圖形。

第二節　監督程式中有關低解像圖形的副程式

在APPLE Ⅱ 的ROM中存在一些彩色低解像圖形的副程式，爲了有效的使用低解像畫面，我們必須了解它們及所使用的一些零頁區位址。

(GBASL，GBASH)	($26，$27)
H2	$ 2C
V2	$ 2D
MASK	$ 2E
COLOR	$ 30

(GBASL，GBASH)指位器指向所將畫的水平線在螢幕上最左邊一點所對應的記憶區位址，再利用6502 組合語言中的(GBASL)，Y索引位址法便可畫出此水平線上各點。COLOR的內含值即爲所欲繪出的顏色，通常設定其值爲由$00，$11至$FF。MASK則用以遮蓋上下相疊方塊中不印出來的方塊，若MASK 的內含值爲$0F，則畫出上方塊(偶數列)，若內含

值爲＄F０則畫出下方塊。而（H２，V２）則分別爲HLIN指令與VLIN 指令畫線時的直線最右與最底端座標。

以下便是一些有關的副程式：

PLOT：＄F800

使用此副程式之前應將X座標值存入Y暫存器，Y座標值存入A暫存器。則此副程式會將＄30位址所指定的顏色顯現在（x，y）螢幕座標上。

PLOT1：＄F80E

這個副程式事實上是PLOT的一部份，若所畫的點與前一次所畫的點Y座標不變，則利用PLOT1，可加快速度。若沒有指定顏色，則使用最近一次所用過的顏色。例如我們想在螢幕上畫出（３，７）及（９，７）兩點：

```
LDY    #$3      ；X座標
LDA    #$7      ；Y座標
JSR    $F800    ；PLOT（３，７）
LDY    #$9      ；新的X座標
JSR    PLOT1    ；PLOT 1（９，７）
```

SETCOL：＄F864

此副程式可用來設定所欲畫出點的顏色，使用此副程式之前只須要將A暫存器的４個低位元存入顏色碼（＄０－＄F），另外４個高位元任何值均沒有影響。

```
如：    LDA     #$6
        JSR     SETCOL    ；$F864
```

便可將顏色設定爲藍色。但除非必要，在設定顏色時，不如自己動手將顏色碼存入＄30中（但顏色碼須由＄00，＄11至＄EE，

$FF）較有效率。如：

```
        LDA   #$66
        STA   COLOR   ; COLOR = $30
```

HLINE：$F819

此副程式與BASIC中的HLINE相同，用來畫水平線。使用前，水平線的Y座標值須存入A暫存器中，左端點的X座標要存入Y暫存器中，右端點的X座標則存入H2中。使用時應特別注意右端的X座標絕不可小於左端的X座標。

VLINE：$F828

此副程式與BASIC中的VLINE相同，用來畫垂直線。使用前，垂直線的X座標值須先存入Y暫存器中，上端點的Y座標要存入A暫存器中，下端點的Y座標則存入V2中。同樣的A暫存器中的值不可大於V2中的值。

SCRN：$F871

此副程式與BASIC中的SCRN（X，Y）相同，用來求出（X，Y）座標點上的顏色碼。使用時將X座標存於Y暫存器中，Y座標存於A暫存器中，才JSR $F871。等到此副程式執行完後，會將0-15的顏色碼存在A暫存器中。

GBASCALC：$F847

此副程式主要是用來求出（GBASL，GBASH）之內含值，使其指向所欲畫出方塊的該列在記憶區的起始位址，使PLOT副程式能利用（GBASL），Y零頁索引位址法存取該列各點的內含值。使用前應將Y座標值除以2後存入A暫存器中（因2n與2n+1列均有相同的（GBASL，GBASH）值，即使用同樣的位元組，只是2n列使用位元組的低半位元組，而2n+1列使用高半位元組），使A暫存器中的值由0至23。

如：欲取出螢幕座標爲（7，26）之點的內含值

```
LDA     #$1A
LSR                     ;除以2
JSR     GBASCALC
IDY     #$7
LDA     (GBASL),Y
```

由以上的討論可知若將LDA #$1A改成LDA #$1B 結果仍然相同，只是偶數（7，26）的顏色碼存於最後A暫存器的0-3位元而（7，27）的顏色碼則存於4-7位元。

SET GR：$FB40

此副程式類似於BASIC中的GR指令。設定低解像圖面，並清出螢幕上面40列低解像畫面。

CLRTOP：$F836

此副程式用以清出螢幕上面40列低解像畫面。通常設定低解像螢幕一般均用JSR $FB40便已足夠，因SETGR 副程式中已呼叫此CLRTOP副程式來做這件工作。

CLRSCR：$F832

此副程式可用以清除所有48列低解像畫面。

最後爲了加深讀者的印像，我們再看一個例子：

```
10  GR : COLOR=6 : PLOT 5,3
20  COLOR=9;HLIN 2,8 AT 6
30  RETURN
```

這是一個用BASIC寫的一個畫圖副程式，將螢幕上的（5，3）點畫成藍色；並用橙色自（2，6）至（8，6）畫出一條水

平線。同樣的程式我們也可以用組合語言來寫：

```
JSR   $FB40     GR
LDA   #$66   ┐
STA   $30    ┘  COLOR=6
LDY   #05    ┐
LDA   #03    │  PLOT 5,3
JSR   $F800  ┘
LDA   #$99   ┐
STA   $30    │
LDA   #$08   │
STA   $2C    │  HLIN 2,8 AT6
LDY   #$02   │
LDA   #$06   │
JSR   $F819  ┘
RTS
```

上面二個副程式作用相同，但用組合語言編寫，顯然痲煩得多。但讀者不要忘了組合語言的好處——速度快。事實上，上面兩個程式在速度上相差了 20 倍之多。

第四章　高解像圖面

在APPLEⅡ提供的二種圖形畫面中，通常大家都較偏愛高解像圖面，因為它能產生較為細膩逼真的影像。高解像畫面每列由280個點構成，一共有192列，使整個銀幕可由280×192=53760個點（低解像圖面僅由1920個點構成）所組成，但僅具有6種不同的顏色。

在開始正式介紹本章之前作者想提醒讀者注意一個細節：在BASIC中，有二個指令HGR與HGR2用來設定高解像圖面第一頁及第二頁，但同時也會將原有的圖形資料清除。可能有時候我們在高解像圖形幕中畫出一些圖形後想暫時回到文字幕顯示一些訊息後再回到高解像圖形幕，此時若使用HGR或HGR2便會使得原來的圖形被清除，碰到這種情形我們通常採用下列方式，避免原有的圖像被清除：

	POKE	−16304，0	設定圖形幕
	POKE	−16297，0	設定為高解像圖形
	POKE	−16300，0	選擇第一頁區
或	POKE	−16299，0	選擇第二頁區

而且，在APPLESOFT中若使用TEXT指令，會將原來的圖形幕轉換至文字幕第一頁區，但Integer BASIC則會根據原來是高解像圖面第一頁或第二頁而轉換至文字幕的第一頁區或第

二頁區。因此往往產生困擾，此時可利用POKE -16300,0以選擇文字幕第一頁區。

第一節　高解像螢幕與記憶區的對應關係

高解像圖形記憶區第一頁及第二頁分別位於＄2000—＄3FFF（8192—16383）和＄4000—＄5FFF（16384—24575）的位址。由於這些位址均固定，因此我們在BASIC程式中若使用到高解像圖面，應特別注意不要使用到此高解像圖形記憶區。一般可將LOMEM：設定在高解像記憶區以上的位置。如欲使用高解像像圖形第一頁則可設定LOMEM：16384。

在正式說明高解像螢幕與記憶區之對應關係前，讀者可先試試下列程式。

```
10  HGR : POKE  -16302 , 0
20  FOR  I = 8092  TO  16384
30  POKE  I , 255
40  NEXT  I  :  END
```

由螢幕上所顯現的結果可以發現，高解像位址與螢幕對應的次序與文字幕的情形差不多。將螢幕分為三部份，第0列畫完後便跳至第二部份的第一列（即64列）去畫，等畫完後再畫第三部份的第一列（128列）。等第4次畫線時便再回到第一部份畫第9條（8列）水平線，如此繼續下去便可將整個螢幕掃瞄完畢，變成全白的螢幕。若讀者在螢幕掃瞄至一半時按下CONTROL-C鍵，使掃瞄停止，便可發現螢幕上的掃瞄線共分為24組（掃瞄結束後，每組由8條平行掃瞄線組成。

事實上，在高解度畫面上，每一位元組是代表高解像度的7個點而不是一個字形，因此在前面所提的螢幕掃瞄程式執行時，各水平掃瞄線並不是一點一點快速掃瞄，而是一次出現7點，40次後完成一條水平掃瞄線（讀者可將30 POKE I，255改成30 POKE I，255：FOR J＝1 TO 100：NEXT J 以便觀察）。由此可知畫面上一列共包括40個位元組280 個點，但每一位元組中的最高位元（bit 7）並不是用來顯示而是做顏色控制之用，故192條水平線共需192×40＝7680個位元組。而每一頁高解像記憶區占8K，因此有8192－7680＝512 個位元組並沒有被使用。表4-1 爲高解像畫面與記憶位址的安排，表中左邊0－191 的數字爲螢幕上的垂直座標（即列數），右邊爲該列之起始位址及結束位址（相差39 個位元組）。此外在表4-1中，我們用粗線條按照螢幕的對應關係將將表分成三部份，此三部份的各對應列（如0、64、128列與8、72、136 列）位址均僅相差40。整個表4-1 ，又分成24小格，每格含8列。在這24個方格的任一方格中，列數差1其位址相差1024 。方格之間的關係則是：一方格與其上或其下之方格，各對應列位址均相差128。經過這樣的安排，我們便可很清楚畫面與記憶位址的對應關係了。

0	8192- 8231	48	8960- 8999	96	8744- 8783	144	8528- 8567
1	9216- 9255	49	9984-10023	97	9768- 9807	145	9552- 9591
2	10240-10279	50	11008-11047	98	10792-10831	146	10576-10615
3	11264-11303	51	12032-12071	99	11816-11855	147	11600-11639
4	12288-12327	52	13056-13095	100	12840-12879	148	12624-12663
5	13312-13351	53	14080-14119	101	13864-13903	149	13648-13687
6	14336-14375	54	15104-15143	102	14888-14927	150	14672-14711
7	15360-15399	55	16128-16167	103	15912-15951	151	15696-15735
8	8320- 8359	56	9088- 9127	104	8872- 8911	152	8656- 8695
9	9344- 9383	57	10112-10151	105	9896- 9935	153	9680- 9719
10	10368-10407	58	11136-11175	106	10920-10959	154	10704-10743
11	11392-11431	59	12160-12199	107	11944-11983	155	11728-11767
12	12416-12455	60	13184-13223	108	12968-13007	156	12752-12791
13	13440-13479	61	14208-14247	109	13992-14031	157	13776-13815
14	14464-14503	62	15232-15271	110	15016-15055	158	14800-14839
15	15488-15527	63	16256-16295	111	16040-16079	159	15824-15863
16	8448- 8487	64	8232- 8271	112	9000- 9039	160	8784- 8823
17	9472- 9511	65	9256- 9295	113	10024-10063	161	9808- 9847
18	10496-10535	66	10280-10319	114	11048-11087	162	10832-10871
19	11520-11559	67	11304-11343	115	12072-12111	163	11856-11895
20	12544-12583	68	12328-12367	116	13096-13135	164	12880-12919
21	13568-13607	69	13352-13391	117	14120-14159	165	13904-13943
22	14592-14631	70	14376-14415	118	15144-15183	166	14928-14967
23	15616-15655	71	15400-15439	119	16168-16207	167	15952-15991
24	8576- 8615	72	8360- 8399	120	9128- 9167	168	8912- 8951
25	9600- 9639	73	9384- 9423	121	10152-10191	169	9936- 9975
26	10624-10663	74	10408-10447	122	11176-11215	170	10960-10999
27	11648-11687	75	11432-11471	123	12200-12239	171	11984-12023
28	12672-12711	76	12456-12495	124	13224-13263	172	13008-13047
29	13696-13735	77	13480-13519	125	14248-14287	173	14032-14071
30	14720-14759	78	14504-14543	126	15272-15311	174	15056-15095
31	15744-15783	79	15528-15567	127	16296-16335	175	16080-16119
32	8704- 8743	80	8488- 8527	128	8272- 8311	176	9040- 9079
33	9728- 9767	81	9512- 9551	129	9296- 9335	177	10064-10103
34	10752-10791	82	10536-10575	130	10320-10359	178	11088-11127
35	11776-11815	83	11560-11599	131	11344-11383	179	12112-12151
36	12800-12839	84	12584-12623	132	12368-12407	180	13136-13175
37	13824-13863	85	13608-13647	133	13392-13431	181	14160-14199
38	14848-14887	86	14632-14671	134	14416-14455	182	15184-15223
39	15872-15911	87	15656-15695	135	15440-15479	183	16208-16247
40	8832- 8871	88	8616- 8655	136	8400- 8439	184	9168- 9207
41	9856- 9895	89	9640- 9679	137	9424- 9463	185	10192-10231
42	10880-10919	90	10664-10703	138	10448-10487	186	11216-11255
43	11904-11943	91	11688-11727	139	11472-11511	187	12240-12279
44	12928-12967	92	12712-12751	140	12496-12535	188	13264-13303
45	13952-13991	93	13736-13775	141	13520-13559	189	14288-14327
46	14976-15015	94	14760-14799	142	14544-14583	190	15312-15351
47	16000-16039	95	15784-15823	143	15568-15607	191	16336-16375

表4-1　高解像畫面與記憶位址的對應關係

由上面表 4-1 的討論，我們導出一個公式，可對任意列，求出其在記憶區的起始位址：首先假設Y爲 0 至 191的列數，由A

＝INT（Y／64），可求出該列位於表4-1中，三大部份的第幾部份；且B＝Y－64＊A　，可求出該列位於此部份的第幾行；C＝INT（B／8）則可求出該列位於此部份的第幾個方格中；又D＝B－C＊8可求出該列位於此方格的第幾列。最後導出：

Y列的起始位址＝8192＊P＋1024＊D＋128＊C＋40＊A
（4－1）式　　P：爲高解像第一頁或第二頁

此外若我們只想求出表4-1中每個方格第一線之位址，則仿照上面的討論可求出下式：（注意：此時Y值爲由0至23）

ADDRESS＝8192＊P＋（Y－INT（Y／8）＊8）＊128
＋INT（Y／8）＊40　（4—2）式

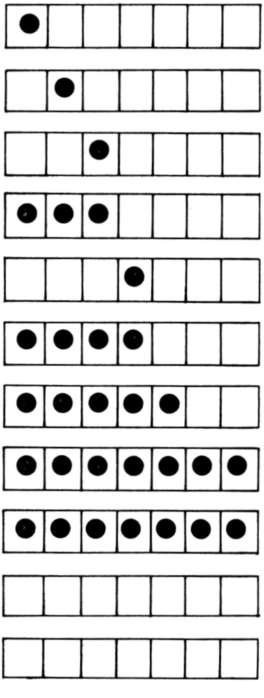
但讀者在使用4－1 式時應注意到，若想求高解像螢幕座標（x，y），0≦x≦279，0≦y≦191，的位址時應將由4-1式所求出的結果加上INT（X／7），因一個位元組控制7點故需將X除以7。至於4-2式，求出表4-1中每一方格的起始位址後，只要在此方格內的其它各列，其差值均爲1024 個位元組，如此便可不必經由複雜的計算而求出其他各列的起始位址。且同在表4-1各方格中的8列，在螢幕上顯示後正好對應於文字幕的一列（即在文字幕上的一列，在高解像螢幕上是由8條水平線所組成）。

利用這個關係，便可以在高解像螢幕上做類似文字幕的圖形安排。如在下面一節的程式中，我們將0至9的字形，顯示在高解像畫面中，但顯示的位置座標（X1，Y1）卻是利用文字幕之第Y1列第X1行，0≦X1≦39，0≦Y1≦23，再經由4-2式轉換成高解像記憶區之位址。請讀者繼續閱讀下一節。

第二節　高解像畫面結構

在前一節中已提過高解像記憶區每一個位元組的 0 至 6 位元，控制螢幕上水平 7 點的顯示或不顯示，位元 7 則做顏色控制之用。只要在高解像記憶區的某一位元組中，0 － 6 位元存在 1 便可在螢幕的對應位置上出現一個亮點，其關係可以下圖表示：

螢幕上“點”的位置

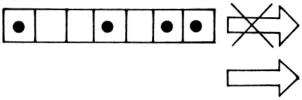


對應之位元組

bit	7	6	5	4	3	2	1	0	
	0	0	0	0	0	0	0	1	$01
	0	0	0	0	0	0	1	0	$02
	0	0	0	0	0	1	0	0	$04
	0	0	0	0	0	1	1	1	$07
	0	0	0	0	1	0	0	0	$08
	0	0	0	0	1	1	1	1	$0F
	0	0	0	1	1	1	1	1	$1F
	0	1	1	1	1	1	1	1	$7F
	1	1	1	1	1	1	1	1	$FF
	1	0	0	0	0	0	0	0	$80
	0	0	0	0	0	0	0	0	$00

圖 4-1

由圖 4-1 可知道，不可直接將螢幕上亮點情形，直接轉換成 2 進位數字，而應注意其次序如：



1001011，$4B

1101001，$69

我們可將圖 4-1 印證一下：先設定HGR後，將 128（$80）及 0，分別 POKE 至 8192 ，發現均沒有顯示亮點，仍為黑色；再將 128（$7F）及 255（$FF）POKE 至 8192，發現均使螢幕在左上角出現 7 個白點，兩者也沒有差異。我們知道這是因為位元 7 只是當顏色控制之用，故不會有額外的亮點。但既然位元 7 是控制顏色，則 $7F 與 $FF 所出現的應該是不同的顏色才對，這也是為什麼 Apple 上所區分出 white 1，white 2 及 black 1 ，black 2 的原因。其差別只是位元 7 為 0 或 1 而已。

我們知道，低解像畫面可利用位元組中的高或低半位元（4 bits）中的 0 至 F 表示出對應方格點的 16 種顏色；而高解像圖面只有六種不同的顏色（黑、白、藍、綠、紫紅及橙色），且一個點只是由一個位元控制其明暗，則其顏色應如何控制呢？

在 Apple Ⅱ 中，任一水平列最左方的點，若其對應的位元組的最高位元為 0 ，則出現紫紅色，下一個點則為綠色，再下面一個點則又變為紫紅色，如此不斷反覆。因此共有 140 對紫紅色和綠色的點，而紫紅色永遠在偶數行（即 0，2，4，……，278），綠色點則永遠在奇數行（即 1，3，5，……，279）。但若各位元組的最高位元為 1 ，則上述的紫紅色與綠色點分別改由藍色與橙色取代。如下圖所示：

(1)各位元組之最高位元均爲 0 （V：紫紅；G：綠）

V	G	V	G	V	G	V	G	V	G	V	G	V	G	V	G	V	……

$2000 第 0 位元組（偶數）　$2001 第 1 位元組（奇數）　$2002 第 2 位元組（偶數）

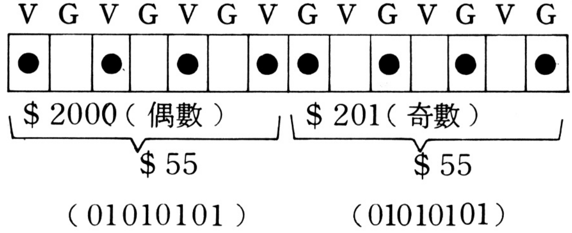
(2)各位元組之最高位元均爲 1 （B：藍；O：橙）

B	O	B	O	B	O	B	O	B	O	B	O	B	O	B	O	B	O	…

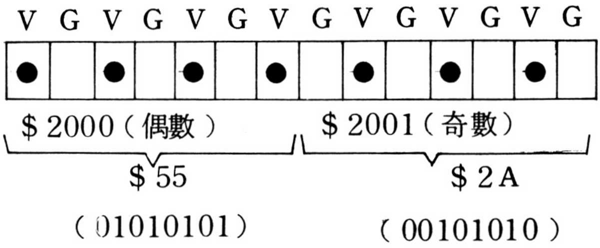
$2000 第 0 位元組（偶數）　$2001 第 1 位元組（奇數）　$2002 第 2 位元組（偶數）

但紫紅色與綠色，藍色與橙色在Apple II 中爲互補色，即若任意相鄰的兩點同時亮，則兩點均成爲白色，因此若銀幕上存在白色以外的平行線則必爲不連續的色線。

此外由上面的說明，我們注意到一件事即：偶數位元組在螢幕上對應的第一點若不是白色則必爲紫紅色或藍色，而奇數位元組則必爲綠色或橙色。因此若我們將 $ 55 POKE 至 $ 2000 可在螢幕上顯示出一條紫紅色的水平線，但若同樣將 $ 55 POKE 至 $ 2001 ，則顯示的欲是一條綠色的水平線，如下圖：



故我們若想使＄2001也出現紫紅色則應將＄2A POKE至＄2001。如下圖：



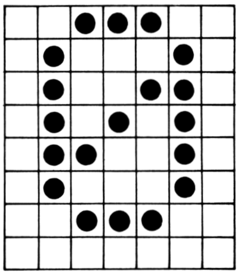
同樣的道理，我們可列出下表

顏　色	偶數位元組	奇數位元組
紫紅色	＄55	＄2A
綠　色	＄2A	＄55
藍　色	＄D5	＄AA
橙　色	＄AA	＄D5
白　1	＄7F	＄7F
白　2	＄FF	＄FF
黑　1	＄00	＄00
黑　2	＄80	＄80

下面我們介紹一個模擬太空飛行的程式，首先程式利用57行在螢幕上畫出一個長方形框當作太空船的監視螢幕，接著60-65行及71-80行畫出兩艘太空船。事實上，是畫出近似半圓形的太空船（第一艘位於螢幕左方，第二艘則位於右方）；程式中的變數Z，是代表圓的弧度（以度爲單位），變數R代表圓的半徑，其中第一艘太空船（60-65行之半圓）是以（10，96）爲中心，而第二艘則是以（275，60）爲中心，大小隨半徑而不

同。程式第 87 行，利用RND 亂數產生函數在螢幕上顯示散亂的 100顆星星。至此整個螢幕感覺上就好像由一艘停在太空中的太空船螢幕上望出去，一片星空和兩艘靜止只看到部份的太空船。

接著程式101-104 行，在螢幕中心點產生 9至 0的數字，代表倒數計時；102行利用ON…GOSUB 指令選取0至9的字形，再用GOSUB 400 將其顯示在螢幕的特定位置。103行使數字顯示後暫停一段時間，再由104行的GOSUB 210 選取空白字形，再用GOSUB 400將此空白字形顯示，其用意即將102行所顯示的數字清除，以便再顯示下一個數字。而事實上這個動作並非必要，沒有這個清除動作，數字顯示的情形，感覺上並不會有什麼差別。0至9的字形存在200至209行，210 行則爲空白字形。例如200行代表0的字形：



A＝	0 0 0 1 1 1 0 0	$ 28
B＝	0 0 1 0 0 0 1 0	$ 34
C＝	0 0 1 1 0 0 1 0	$ 50
D＝	0 0 1 0 1 0 1 0	$ 42
E＝	0 0 1 0 0 1 1 0	$ 38
F＝	0 0 1 0 0 0 1 0	$ 34
G＝	0 0 0 1 1 1 0 0	$ 28
	0 0 0 0 0 0 0 0	$ 00

程式400-410的副程式，是利用上一節所討論的4-2 式，求出第一條水平字形掃描線在記憶體之起始位址，而以下7條水平字形掃描線只須重覆加1024 即可。但正如上一節中4-2 式所討論的X 1，Y 1位置座標應採用文字幕之螢幕座標 $0 \leq X1 \leq 39$，$0 \leq Y1 \leq 23$。

最後，100-150 行畫出4顆星星和一個方形框，逐漸向螢

幕逼近。其中A／B，C／D，E／F，G／H 分別代表四顆星星的位置，它們由螢幕的中心位置逐漸擴大。

123行畫出邊長爲P的小正方形，由122及124行檢查是否已擴大至最大範圍。126行主要用來消除原來畫出的正方形，第140行作用也是用來清除原來畫出的四顆星星。

這個程式很簡單，卻能產生很好的視覺效果，讀者何不仿照這種方法展開自己的太空旅程呢！

```
]FPP
]LIST

1  PRINT ""
55  HGR : HCOLOR= 3: HOME : VTAB
     23:
56  FLASH : PRINT "DRAW THE FIRST
      PLANET!"
57  HPLOT 9,0 TO 276,0 TO 276,159
      TO 9,159 TO 9,0
60  FOR Z = 1 TO 180
62 R = 25:X = 10 +  SIN (Z / 180 *
     3.1415) * R
65 Y = 96 -  COS (Z / 180 * 3.141
     5) * R: HPLOT X,Y: NEXT Z
70  VTAB 23: PRINT "NOW THE SECON
     D PLANET!"
71  FOR Z = 180 TO 360
75 R = 20:X = 275 +  SIN (Z / 180
      * 3.1415) * R
80 Y = 60 -  COS (Z / 180 * 3.141
     5) * R: HPLOT X,Y: NEXT Z
85  HOME : VTAB 23: PRINT "NOW AD
     D A FEW STARTS!"
87  FOR I = 1 TO 100:X =  RND (1)
      * 260 + 10:Y =  RND (1) * 1
     40 + 5: HPLOT X,Y: NEXT I
100  HOME : VTAB 23: PRINT "NOW T
     HE STARTS AND SPACE PANEL ZO
     OM IN!": NORMAL
```

```
101  FOR H1 = 10 TO 1 STEP  - 1
102  ON H1 GOSUB 200,201,202,203,
     204,205,206,207,208,209: GOSUB
     400
103  FOR Q1 = 0 TO 1000: NEXT Q1
104  GOSUB 210: GOSUB 400: NEXT H
     1
105 X = 138:Y = 77:J = 1:Z = 3
110 A = 135:B = 75:C = 145:D = 75
     :E = 145:F = 85:G = 135:H =
     85
120  HCOLOR= 7: HPLOT A,B: HPLOT
     C,D: HPLOT E,F: HPLOT G,H
121 L = X:M = Y:K = P:X = X - 2:Y
      = Y - 2:J = J + 2
122  IF J = 75 THEN 124
123 P = 2 * J: HCOLOR= Z: HPLOT X
     ,Y TO X + P,Y TO X + P,Y + P
      TO X,Y + P TO X,Y
124  IF J = 75 THEN J = 1:X = 138
     :Y = 77:P = 2:Z =  RND (1) *
     7 + 1
126  HCOLOR= 0: HPLOT L,M TO L +
     K,M TO L + K,M + K TO L,M +
     K TO L,M
130 A = A - 7:B = B - 5:C = C + 7
     :D = D - 5:E = E + 7:F = F +
     5:G = G - 7:H = H + 5
135 N = N + 1
140  HCOLOR= 0: HPLOT A + 7,B + 5
     : HPLOT C - 7,D + 5: HPLOT E
      - 7,F - 5: HPLOT G + 7,H -
     5
145  IF N = 15 THEN N = 0: HPLOT
     A,B: HPLOT C,D: HPLOT E,F: HPLOT
     G,H: GOTO 110
150  GOTO 120
200 A1 = 28:B1 = 34:C1 = 50:D1 =
     42:E1 = 38:F1 = B1:G1 = A1: RETURN
201 A1 = 8:B1 = 12:C1 = A1:D1 = A
     1:E1 = A1:F1 = A1:G1 = 28: RETURN

202 A1 = 28:B1 = 34:C1 = 32:D1 =
     24:E1 = 4:F1 = 2:G1 = 62: RETURN
```

```
203 A1 = 62:B1 = 32:C1 = 16:D1 =
     24:E1 = B1:F1 = 34:G1 = A1: RETURN
204 A1 = 16:B1 = 24:C1 = 20:D1 =
     18:E1 = 62:F1 = A1:G1 = A1: RETURN
205 A1 = 62:B1 = 2:C1 = 30:D1 = 3
     2:E1 = D:F1 = 34:G1 = 28: RETURN

206 A1 = 56:B1 = 4:C1 = 2:D1 = 30
     :E1 = 34:F1 = E1:G1 = 28: RETURN

207 A1 = 62:B1 = 32:C1 = 16:D1 =
     8:E1 = 4:F1 = E1:G1 = E1: RETURN

208 A1 = 28:B1 = 34:C1 = B1:D1 =
     A1:E1 = B1:F1 = B1:G1 = A1: RETURN
209 A1 = 28:B1 = 34:C1 = B1:D1 =
     60:E1 = 32:F1 = 16:G1 = 14: RETURN
210 A1 = 0:B1 = A1:C1 = A1:D1 = A
     1:E1 = A1:F1 = A1:G1 = A1: RETURN

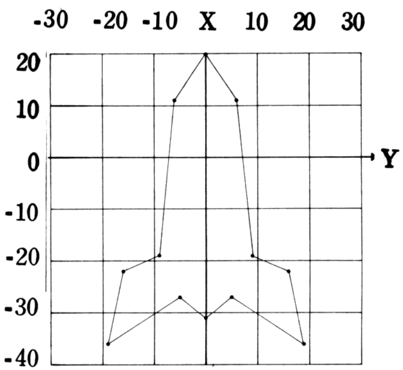
400 Y1 = 10:X1 = 19
410 W = Y1 -  INT (Y1 / 8) * 8:W1
      = 8192 + W * 128 +  INT (Y1
      / 8) * 40 + X1: POKE W1,A1:
      POKE W1 + 1024,B1: POKE W1 +
     2048,C1: POKE W1 + 3072,D1: POKE
     W1 + 4096,E1
411  POKE W1 + 5120,F1: POKE W1 +
     6144,G1: POKE W1 + 7160,0: RETURN
```

第三節　圖形的製作

在前一節的程式中，我們利用高解像畫圖指令HPLOT X1，Y1 TO X2，Y2〔{TO X_n，Y_n}〕畫正方形。讀者可能會產生一個疑問：若圖形不是正方形，而是較複雜的圖形，則使用

HPLOT指令豈非很不方便？對於這個問題我們可使用一點簡單的技巧，使HPLOT用起來較方便一點，下面便是一個例子。

在後面的程式中，主要是在高解像螢幕上劃出一艘火箭，由螢幕底端向上飛行，到達螢幕頂端而消失。在此程式中我們利用HPLOT指令劃出火箭圖形。圖 4-2 便是火箭的座標圖形，但此座標爲相對於原點（0，0）的座標，並非螢幕上的絕對座標，利用這個方式我們只須改變原點之位置便可將圖形畫在螢幕上的任何位置（只要不超過螢幕範圍）



點	X	;	Y
1	0		20
2	6		11
3	9		-19
4	16		-22
5	19		-36
6	5		-27
7	0		-31
8	- 5		-27
9	-19		-36
10	-16		-22
11	- 9		-19
12	- 6		11
13	0		20

圖 4 - 2

在程式 1010 行的DATA中將 1 至 13 點的座標順序存入（第 1 和 13 點重覆，以完成整個封閉曲線）而由第 45 行將X，Y座標值順序存入X，Y 陣列中。此外在程式第 30 行須先讀入預先在 1000 行所設定的點數（因在BASIC中由 0 算起，故雖有 13 點但設定N＝12）。第 10 行的POKE - 16302， 0是設定高解像全螢幕。

程式第 50 及 55 行利用FOR…NEXT 迴圈，設定圖形中心

位置，並不斷改變Y座標值使圖形向上移動。在60行利用HCOLOR＝3畫出圖形後，65行暫停，70行再利用HCOLOR＝4將圖形消除。眞正畫圖則在500行的副程式中畫出。

```
10  HGR : POKE  - 16302,0: REM  F
     ULL SCREEN
20  READ N
40  DIM X(N),Y(N)
45  FOR I = 0 TO N: READ X(I),Y(I
     ): NEXT I
50 I = 140
55  FOR J = 40 TO 170 STEP 5
60  HCOLOR= 3: GOSUB 500
65  FOR Z = 0 TO 100: NEXT Z
70  HCOLOR= 4: GOSUB 500
80  NEXT J
90  END
500  HPLOT I + X(0),J + Y(0): FOR
     K = 1 TO N: HPLOT  TO I + X(
     K),J + Y(K): NEXT K: RETURN

1000  DATA  12
1010  DATA  0,20,6,11,9,-19,16,-2
     2,19,-36;5,-27,0,-31,-5,-27,
     -19,-36,-16,-22,-9,-19,-6,11
     ,0,20
```

上述方法，可使HPLOT指令用起來較靈活，但我們發現圖形較複雜時（如上述之火箭）執行的速度就不夠快了，同時圖形的放大、旋轉均很不容易。因此在一般GAME 程式中，並不常用這個方法，而採用圖形表（shape table）的方法。

圖形表的功能是APPLESOFT BASIC的一項特色，主要是利用一些向量來組成圖形，圖形造成後可畫在螢幕的任何位置

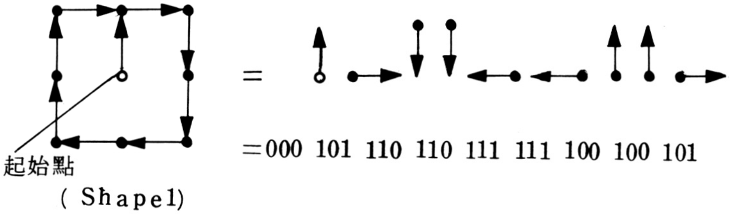
，並可旋轉，放大，消除圖形時亦不會破壞襯有底色的背景圖面。

製作圖形表時，首先將欲繪出的圖形在繪圖紙上選定一個起始點將圖形以上、下、左、右的箭頭向量依序組成。此箭頭向量代表高解像螢幕上一單位座標的改變，因此上、下、左、右便有四種箭頭向量，再加上畫點或不畫點之移動便有8種箭頭向量，各有其代碼如下：

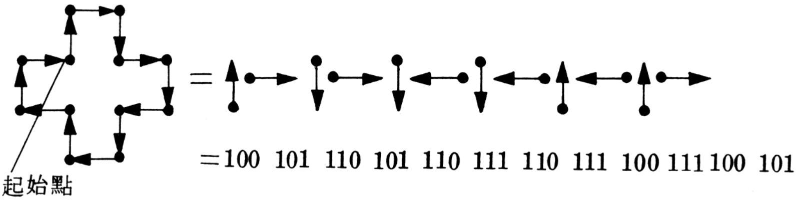
向量	作　用	二進位代號	向量	作　用	二進位代號
↑	不畫點向上移動	000	↑	畫點後向上移動	100
→	不畫點向右移動	001	→	畫點後向右移動	101
↓	不畫點向下移動	010	↓	畫點後向下移動	110
←	不畫點向左移動	011	←	畫點後向左移動	111

表 4-2

將圖形之向量圖完成後，再將圖形分解，按表 4-2 將各向量化成二進位代碼，如下所示：



=000 101 110 110 111 111 100 100 101



(Shape 2)　　**圖 4-3**

接下來的工作便是將各向量代碼組成 16 進位數字存入圖形表中，而組合過程是依下述的方法進行。

圖形表的每一個位元組可分爲三部份，第一、二部份各爲 3 個元位可存入任意向量代碼。但第三部份，只包括兩個位元，故只能存入部份向量代碼，即 （010），（001），（011）三種，即不畫點移動之向量代碼（因其最高位元均爲 0）；但第三部份亦可不使用即存入 00。讀者應注意到第三部份的 00 並非向量 （000）之代碼，而是代表不使用。意即我們不能將向量 （000）的代碼存入第三部份中，否則 Apple 會當成不使用，而使得圖形表中失去“不畫點向上移動”的向量。

位元	7	6	5	4	3	2	1	0
	1	0	2	1	0	2	1	0
	M	M	P	M	M	P	M	M
	第三部份		第二部份			第一部份		

P：代表移動前，要**不要畫點，0 不畫，1 畫**

MM：代表移動的方向

0 0	向上移	0 1	向右移
1 0	向下移	1 1	向左移

圖 4-4

由圖4-4 我們知道表4-2 各代碼事實上便是圖4-4中PMM對之組合。

當三部份均爲0時，Applesoft 認爲已到達圖形表之結束位置（由此可知圖形表的最後一個位元組須存入0代表圖形表的結束），因此限制了↑不畫點向上移動的箭頭向量的使用。例如若第一、二部份均已存入不畫點向上移動之向量代碼後，第三部份便不能再使用↑非得使用↓、→、←中的任一個向量。因此若需同時使用數個不畫點向上移動之向量時，我們可先使用一次↑，然後利用←向左移動後，再使用→向右移回原位，然後才再使用一次 ↑ 向量，如此反覆便可達到目的。

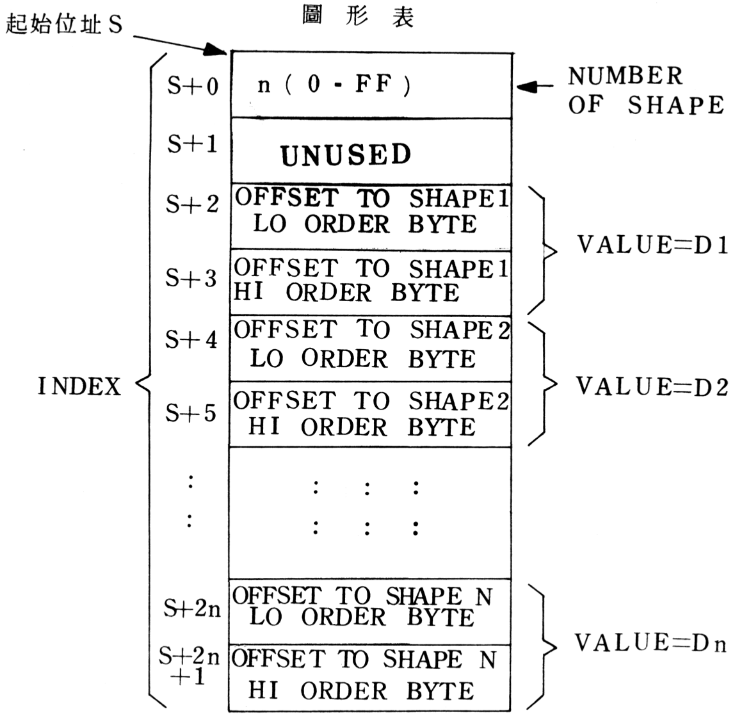
接著我們便可將圖4-3的各向量代碼，依上述方法轉換成16進位。如下所示：

	00 101 000	0010 1000	$ 28	
shape 1	00 110 110	0011 0110	$ 36	
	00 111 111	0011 1111	$ 3F	
	00 100 100	0010 0100	$ 24	
	00 000 101	0000 0101	$ 05	
	00 000 000	0000 0000	$ 00	→ 結束

	00 101 100	0010 1100	$ 2C	
	00 101 110	0010 1110	$ 2E	
shape 2	00 111 110	0011 1110	$ 3E	
	00 111 110	0011 1110	$ 3E	
	00 111 100	0011 1100	$ 3C	
	00 101 100	0010 1100	$ 2C	
	00 000 000	0000 0000	$ 00	→ 結束

接下來的步驟便是如何將圖形之16進位值存入圖形表中。在正式存入之前，首先讓我們了解整個圖形表的結構，如表4-3。

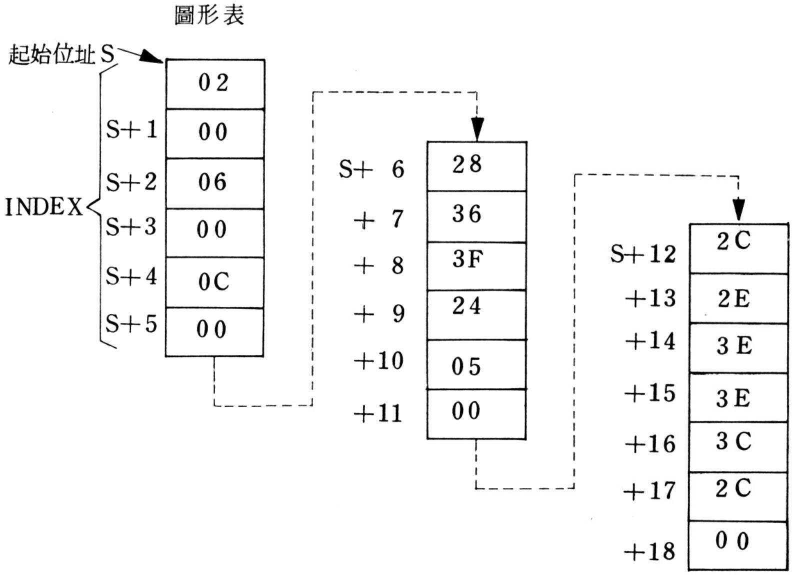
由表4-3中我們可了解整個圖形表的構成，但在此要提醒讀者注意的是：雖然在建立圖形表時，已根據實際圖形的個數，定義記憶空間，但Applesoft並沒有硬性規定圖形的16進位值必須存入圖形表中，因此通常我們可在圖形表中預留位置，以便將來能再加入新的圖形；否則往後想在已定義完整的圖形表中加入新的圖形，便很麻煩了。



S+D1	FIRST BYTE	shape 1 PLOTTING VECTOR
	: : : : : :	
	LAST BYTE=00	
S+D2	FIRST BYTE	shape 2 plotting vector
	: : : : : :	
	LAST BYTE=00	
	: : : : : :	
S+Dn	FIRST BYTE	shape n plotting vector
	: : :	
	LAST BYTE=00	

表 4-3

最後我們便可根據表 4-3 將以上所建立之二個圖形存入圖形表中。完成後整個圖形表成為：



圖形表完成後，在使用前應先將圖形表存入記憶體中，並將圖形表的起始位址存入 $E8（低位元組）及 $E9（高位元組）中。

第四節　圖形表的使用

利用圖形表在螢幕上繪圖，是APPLESOFT的GRAPHIC GAME 中最主要的繪圖方式，因此我們除了要會建立圖形表，還要熟悉其應用方式。所以在本節中，將介紹使用圖形表的幾個重要指令，及三個實際的例子，供讀者與上一節的火箭飛行程式做比較。

DRAW N AT X，Y 是將圖形表中所定義的第N個圖形顯示在高解像螢幕座標（X，Y）上。此（X，Y）即開始畫第一個

點的位置（不論此點是否繪出）；而XDRAW N AT X，Y 是用來將螢幕上先前的圖形消除而又不會影響背景的顏色。其方式是將圖形與螢幕上原先的圖形做EXCLUSIVE-OR，因此若先前螢幕上沒有圖形則XDRAW的作用便會同DRAW 一樣將圖形顯示在螢幕上。原則上，雖然DRAW 指令所繪出的圖形可用XDRAW 消除，但我們發現在某些情況中（即在某些ROT 指令所定義的旋轉角及SCALE所定義的圖形大小）XDRAW指令無法將DRAW 指令所繪出的圖形消除，故作者建議大家不妨用XDRAW指令取代 DRAW 指令，無論畫圖形或消除圖形均使用XDRAW 指令。

ROT 指令是用來將圖形旋轉一個指定的角度；而SCALE指令是用來定義圖形的放大程度。在第一次用DRAW或XDRAW指令畫圖時，兩者均須事先定義好，否則便無法畫出圖形。SCALE＝1時所畫出的便是原來圖形的大小，ROT＝0 即圖形旋轉0度。

雖然，ROT所定義的值可自0至63。但事實上眞正的旋轉角度與SCALE所定義的值有關。當SCALE＝1時只能旋轉0°，90°，180°及270°四個方向（即ROT＝0，16，32，48）。當SCALE＝2 時則可旋轉8個方向即0°，45°，90°，135°，180°，225°，270°，315°（ROT＝0，8，16，24，32，40，48，56），當SCALE 之值大於或等於5時，便可旋轉全部的64 個方向了。此外圖形旋轉時是根據圖形的第一個畫點（此點可能不顯示）位置旋轉，並非外顯圖形的中心位置，因此我們若想使圖形按圖形的中心位置旋轉，在建圖形表時便須將第一個箭頭向量位置自圖形的中心位置開始。

最後要注意的便是圖形表的儲存位置了。如果讀者已了解第

二章，相信這便不是問題了。如圖形不大時可存入＄300－＄3CF（＄3D0－＄3FF由Apple 磁碟作業系統使用），或將HIMEM 往下移，空出最高位址存圖形表。當然在正式利用圖形表繪圖時不可忘了將圖形表的起始位址存入＄E8（低位元組）及＄E9（高位元組中）。

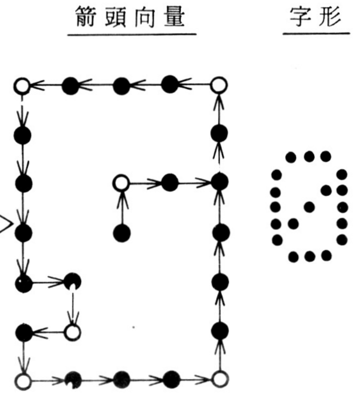
以下我們實際運用上面所介紹的方法，看看下面三個程式：

程式一：

在上一節我們的太空飛行程式中，順便介紹了在高解像畫面上顯示數字的方法。在本程式中，我們將使用不同的方法在高解像畫面上顯示數字。執行過本程式後讀者將會發現幾個特點：程式短，速度快，使用方便可將任何長度的文數字顯示在螢幕上的任意位置。

程式首先GOSUB 10000將HIMEM：指標向上移動，並將圖形表存入HIMEM：以上的記憶區，以免被BASIC程式所破壞。10010至10035行存的是數字0－9的圖形表，如圖形1定義數字0的16進位箭頭向量代碼，其組成方式如下：

	10進位	2進位
	12	00 001 100
10015	37	00 100 101
行	28	00 011 100
DATA	63	00 111 111
	23	00 010 111
	54	00 110 110
	46	00 101 110
	30	00 011 110
	14	00 001 110
	45	00 101 101
	5	00 000 101
	36	00 100 100
	4	00 000 100



第9900—9910行爲顯示數字字形的副程式，將存在變數N中的數字改爲字串後，逐次取出字元顯示。第1005 行利用巢狀迴路（NEST-LOOP）求出X·Y的數值存入N中，而由9900行的副程式顯示。在本程式中第1005行，及9910行DRAW指令所設定的螢幕位置，讀者可根據本身的需要加以修改。此外若想在高解像螢幕上顯示其他的ASCII字元，亦可模仿本節的方法，建立自己的圖形表，再畫出字元圖形即可。

```
]LIST

10  REM  ***************************
11  REM  **  HI-RES NUMBERS       **
12  REM  **  SHAPE TABLE METHOD   **
13  REM  ***************************
100  REM  LOAD 10000-10033 ROUTINE THEN GOSUB IT AT START OF PROGRAM
150  REM  GOSUB 9900 TO PRINT NUMBER
160  REM  N IS NUMBER TO BE DRAW
170  REM  X,Y IS LOCATION TO DRAW NUMBER
200  REM  CHARTERS ARE 0 THRU 9
300  REM  HOWEVER THEY ARE LISTED IN THE SHAPE TABLE THRU 1 THRU 10
400  REM  TO DRAW THEM USE FORMAT 'DRAW N+1 AT X,Y'
1000  GOSUB 10000
1005  HGR : FOR X = 10 TO 260 STEP 45: FOR Y = 5 TO 155 STEP 10:N = X * Y:
      GOSUB 9900: NEXT Y,X: END
9900  HCOLOR= 3: SCALE= 1: ROT= 1:N$ =  STR$ (N): FOR J = 1 TO  LEN (N$)
9910  DRAW  VAL ( MID$ (N$,J,1)) + 1 AT (X - 7) + (J * 7),Y: NEXT J: RETURN

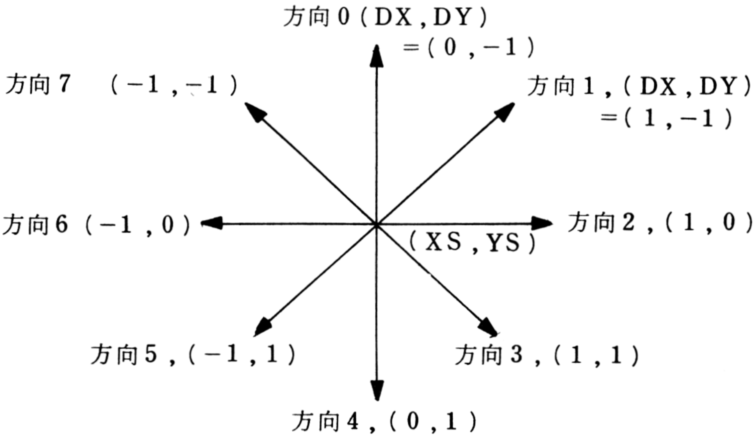
10000  HIMEM: 37888
10005  POKE 232,0: POKE 233,148: FOR I = 37888 TO 38017: READ J: POKE I,J:
      NEXT : RETURN
10010  DATA  10,0,22,0,36,0,44,0,54,0,66,0,76,0,88,0,99,0,108,0,119,0
10015  DATA   12,37,28,63,23,54,46            ,30,14,45,5,36,4,0,36,188, 1
             50,18,45,28,36,0
10020  DATA  101,228,63,23,150,241,46,45,37,0,12,12,60,63,183,146,21,45,12
     ,228,7,0
10025  DATA  58,39,12,12,12,54,174,55,62,0,56,39,44,45,245,170,54,23,63,28
     ,4,0
10030  DATA  117,246,63,28,36,229,12,12,45,6,0,12,12,60,63,183,82,30,46,0
10035  DATA  231,100,45,21,246,14,246,63,28,36,0,231,100,45,21,54,119,30,3
     0,63,4,0
```

程式二：

本程式是模擬GAME 程式中火箭的飛行，但本程式中火箭的飛行與上一節的火箭飛行不同，必須加油（按PUSHBUTT-ON #0）才能移動，且一停止加油火箭便會停止飛行。在火箭

停止或行進中，可任意由PADDLE＃0控制火箭的前進方向（只有8個方向）。若火箭行進至螢幕頂端後可再由螢幕底端出現，由螢幕左（右）側消失後可再由螢幕右（左）側出現，以模擬螢幕的移動。

程式中火箭有8個飛行方向，如下圖：

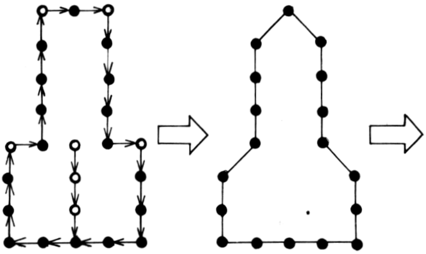


其中（XS，YS）爲圖形在螢幕上的位置；（DX，DY）表示各個方向對目前（XS，YS）的變量，由於8個方向的變量各不相同，我們將每一個方向所對應的變量存入陣列DX（0-7）與DY（0-7）中（見程式220行及260行），因此若想找出火箭的下一個位置，只須根據其方向（由變數TRT代表方向0-7）便可由程式中第80行求出。

由於我們用PADDLE控制0-7的方向，因此必須將輸入的PADDLE值0-255轉換成0-7，此動作由程式第50行完成。此外，程式中利用ROT指令控制圖形的旋轉，由於有八個方向

，故SCALE必須指定為2，且將0-7 方向所對應的ROT 值存入陣列R（0-7）中（見程式230行及270行）。

火箭圖形及圖形表如下圖：（程式中，圖形表被轉換成10進位在250行的DATA 中指明，並由200行讀入存於＄300至＄30F中）。



10	010 010	$92	(146)
	111 111	$3F	(63)
	100 100	$24	(36)
	001 100	$0C	(12)
	100 100	$24	(36)
01	100 100	$64	(100)
	010 101	$15	(21)
00	110 110	$36	(54)
10	101 110	$AE	(174)
	110 110	$36	(54)
	111 111	$3F	(63)

由於本程式的圖形表很小，故將它存在第三頁記憶區中。但一般的GAME 程式喜歡將發聲音的副程式（後面說明），存入此區域，因此使用時應注意。

有些讀者可能認為火箭圖形中的頭三個“不畫點向下移動”的箭頭向量沒有作用。事實上我們使用這三個箭頭向量的原因是因為程式中我們利用ROT = 2 使圖形旋轉，且要按圖形中心旋轉，故加入此三個箭頭向量，定出圖形的中心點。諸入此類，讀者在製作自己的圖形表時均應多加注意。

```
5  GOSUB 200
10  HGR : HCOLOR= 3:XS = 140:YS = 96: POKE  - 16302,0
20  SCALE= 2: ROT= 0:VX = VY = 0
30  XDRAW 1 AT XS,YS
40  FOR Z = 0 TO 100: NEXT Z
45  XDRAW 1 AT XS,YS
50 TRT =  INT ( PDL (0) / 32)
60  ROT= R(TRT)
70 BUT =  PEEK ( - 16287): IF BUT <  = 127 THEN 90
80 XS = XS + DX(TRT):YS = YS + DY(TRT)
82  IF XS > 269 THEN XS = 10: GOTO 86
84  IF XS < 10 THEN XS = 269
86  IF YS > 181 THEN YS = 10: GOTO 90
88  IF YS < 10 THEN YS = 181
90  XDRAW 1 AT XS.YS
100  FOR Z = 0 TO 100: NEXT Z
110  GOTO 45
200  FOR S = 0 TO 15: READ SH: POKE 768 + S,SH: NEXT S
210  POKE 232,0: POKE 233,3
220  FOR I = 0 TO 7: READ DX(I),DY(I): NEXT I
230  FOR I = 0 TO 7: READ R(I): NEXT I
240  RETURN
250  DATA   1,0,4,0,146,63,36,20,36,36,21,54,46,54,63,0
260  DATA  0,-1,1,-1,1,0,1,1,0,1,-1,1,-1,0,-1,-1
270  DATA  0,8,16,24,32,40,48,56
```

程式三：

本程式的重點在於它所利用到的觀念，爲了簡單我們還是模擬飛機的飛行，只是所用的方法與前面所提的方法極不相同。在此我們利用到高解像圖形的第一頁及第二頁，藉著輪流快速的顯示此二頁區的圖形，達到飛機移動的目的。

程式首先清除二頁區的高低解像幕，並設定第二頁爲顯示幕（第 10 行）接著將圖形畫在第二頁區的起始位置上，並求出下一個畫圖位置（ 30-35 行）；與原位置相差平行的五個單位；此時看到的是高解像圖形第二頁，同時我們將新的圖形畫在第一頁的新位置，畫好後便將第一頁圖形顯示出來（ 70 行），並藉著使

用者觀看第一頁圖形之時，將第二頁的原來圖形清除，在新的位置畫出圖形，畫好後再顯示第二頁圖形，再清除第一頁原有圖形畫出新的圖形，如此不斷反覆，直到飛機到達螢幕邊緣。

由此程式讀者可能看不出這種方法的優點。若整個螢幕的圖形很複雜，每次畫好後顯示一下，再將原來圖形清除，再畫上新的圖形，再清除再畫……，這一連串動作顯得很笨拙。若採用本程式中第一頁、第二頁高解像幕交換顯示的方法，可使在顯示螢幕圖形之時，在觀察者感覺不到的情況下，清除原來圖形再畫上新的圖形，再顯示新的圖形，如此可使觀察者無法發覺畫圖和清除圖形的動作，而感覺到好像看幻燈片一般，所看到的是一張張連續的圖片，可節省許多時間。

程式中所用到的位址230（$E6），是Applesoft所使用的一個零頁指標，用以說明圖形是畫在高解像螢幕的第幾頁區，其內含值若為＄20 則代表第一頁，內含值為＄40 代表第二頁。因此若POKE - 16300，0及POKE 230，64， 便可使螢幕呈現高解像第一頁的圖形，而在高解像第二頁繪製新的圖形。

程式中沒有包括圖形表，因此在正式執行此程式前，須先存入下列圖形表：

```
]CALL-151
*300: 01  00  04  00  24  24  4C  12
      15  15  15  2D  2D  2D  2D  6D
      AA  15  3F  3F  3F  3F  3F  3F
      3F  3F  3F  24  24  00  00
*E8 : 00  03
```

將圖形表存入第三頁區＄300，並將圖形表起始位置＄300存入＄E8及＄E9中。

```
]LIST

5 X2 = 30:Y2 = 60
10  HOME : HGR : HGR2 : HCOLOR= 3
15  POKE 230,64
20  ROT= 1: SCALE= 2
30  XDRAW 1 AT X2,Y2
35 I2 = X2:J2 = Y2:X1 = X2 + 5:Y1 = Y2
40  POKE 230,32
45  XDRAW 1 AT X1,Y1
50 I1 = X1:J1 = Y1:X2 = X1 + 5:Y2 = Y1
70  POKE  - 16300,0: POKE  - 16302,0
80  POKE 230,64
90  XDRAW 1 AT I2,J2
95  XDRAW 1 AT X2,Y2
100 I2 = X2:J2 = Y2:X1 = X2 + 5:Y1 = Y2
105  IF X1 >  = 250 THEN  END
120  POKE  - 16299,0: POKE 230,32
130  XDRAW 1 AT I1,J1
135  XDRAW 1 AT X1,Y1
140 I1 = X1:J1 = Y1:X2 = X1 + 5:Y2 = Y1
150  IF X2 >  = 250 THEN  END
160  GOTO 70
```

第五節　APPLESOFT中的幾個高解像繪圖副程式

APPLESOFT ROM中有一些高解像繪圖的副程式，供APPLESOFT BASIC使用，但由於APPLESOFT為解譯程式，故使用起來較慢，若能直接使用組合語言（機器語言）呼叫這些副程式則可使程式的執行速度加快4至6倍。因此在本節中我們搜集幾個較常用到的副程式，提供給讀者做參考。讀者可利

用這些副程式寫一個組合語言（或機器語言）的副程式，供APPLESOFT BASIC的主程式呼叫，以加快執行的速度。在此我們所介紹的幾個高解像繪圖副程式，偏重於使用性，想對這方面資料多了解一點的讀者可參閱參考書目中的Ref 5。

1. HGR：$F3E2

此副程式相當於APPLESOFT中的HGR 指令，不需任何輸入參數，可直接呼叫。顯示高解像第一頁（將$E6 存入$20見上一節），設定圖形文字混合幕。

2. HGR2：$F3D8

顯示高解像第二頁（將$E6存入$40），設定圖形文字混合幕，並清除畫面爲黑色。亦不需任何輸入參數，可直接呼叫。

3. BKGND：$F3F4

清除高解像第一頁或第二頁——根據$E6中之値（$20代表第一頁，$40代表第二頁）。在呼叫前應將底色存入A－暫存器中，此底色乃根據APPLESOFT的COLOR MASKING TABLE而來，其値如下：

HCOLOR = 0時	對應於	$ 00	= 00000000
HCOLOR = 1時	對應於	$ 2A	= 00101010
HCOLOR = 2時	對應於	$ 55	= 01010101
HCOLOR = 3時	對應於	$ 7F	= 01111111
HCOLOR = 4時	對應於	$ 80	= 10000000
HCOLOR = 5時	對應於	$ AA	= 10101010
HCOLOR = 6時	對應於	$ D5	= 11010101
HCOLOR = 7時	對應於	$ FF	= 11111111

如將$ 2A存入A一暫存器中，再呼叫BKGND（當然在此之前

應已設定高解像頁數），便能將高解像螢幕清除爲綠色。

4. HCOLOR：$F6F0

相當於APPLESOFT的HCOLOR 指令，在呼叫前應將0至7的顏色值存入X－暫存器中。此副程式會自動在COLOR MASKING TABLE中找出對應的顏色值。

5. HPLOT：$F457

在呼叫前應將A－暫存器存入螢幕的垂直座標，Y－暫存器與X－暫存器分別存入螢幕的水平高低座標位元組；如螢幕橫座標爲270則須將$1存入Y－暫存器，$0E存入X－暫存器中(因270＝$10E）。利用此副程式可在指定的螢幕座標繪出一點。

6. HLINE：$F53A

在呼叫前，應將螢幕的垂直座標存入Y－暫存器中，螢幕的水平座標高、低位元組分別存入X－暫存器及A－暫存器中。此副程式相似於APPLESOFT BASIC中的HPLOT TO 指令，由先前的畫點位置，畫出一直線直到目前所指定的位置。

7. HPOSN：$F411

呼叫此副程式之前，應將螢幕橫座標之高、低位元組分別存入Y與X一暫存器中，垂直座標值存入A一暫存器中。此副程式會進行一些內部的儲存，通常對此副程式的呼叫，是在執行“DRAW（或XDRAW）”動作之前，先呼叫此副程式，再呼叫DRAW的副程式。

8. SHPTR：$F730

從上一個副程式，到副程式10均爲針對BASIC 中的DRAW或XDRAW指令。此副程式用來指出欲畫出的圖形是圖形表的第幾個圖形。如爲第n 個圖形，則將n存入X－暫存器中，再呼叫此副程式即可。此副程式會將位址$1A，$1B存入此圖形

之位址（低、高位元組指標）。

9. DRAW：$F601

此副程式眞正進行繪圖動作，但在呼叫此副程式之前應先將“ROT=”（旋轉方向值）之值存入A－暫存器中，並將$1A及$1B之內容取出分別存入X－暫存器與Y－暫存器中。

10. XDRAW：$F65D

此副程式可用以消除螢幕上的圖形，與BASIC中的XDRAW意義相同，但呼叫前，應如副程式9所指出者設定好各輸入參數。

此外，我們已經知道在BASIC中若想利用DRAW或XDRAW指令畫出圖形表中的圖形，在DRAW(或XDRAW)指令之前應先將圖形表起始位置存入$E8及$E9，並設定ROT及SCALE之值，然後才能DRAW n AT X，Y。（當然HGR與HCOLOR亦須先定好）。但讀者可能已發現，在上面的討論中均沒有提到SCALE之設定。事實上在組合語言的情況中，SCALE的設定與ROT不同，ROT是在呼叫DRAW之前將其值存入A－暫存器中，但SCALE之值卻必須在此之前先存入一個記憶區位址$E7之中。即$E7內存SCALE FACTOR，而$E8及$E9內存圖形表之（低、高位元組）起始位址。

綜合以上的說明，假設我們已有一個圖形表存在記憶區中起始位址爲$2000，現在我們想將第一個圖形畫在螢幕上X＝64，Y＝64的位置，顏色爲白色，SCALE＝2，ROT＝0，則可以下列組合語言表出：

```
        ⋮
   JSR      HGR            ; HGR : $F3E2
```

```
LDA     #$00     ;
STA     $E8      ;存入圖形表
LDA     #$20     ;之起始位置
STA     $E9      ;
LDX     #$03     ;白色
JSR     HCOLOR   ; HCOLOR = 3
LDA     #$02     ;
STA     $E7      ; SCALE = 2
LDX     #$01     ;圖形# 1
JSR     SHPTR    ;設定圖形 1 的指標
LDX     #$40     ;
LDY     #$00     ; X = 64
LDA     #$64     ; Y = 64
JSR     HPOSN
LDX     $1A      ;
LDY     $1B      ;圖形位址
LDA     #$00     ; ROT = 0
JSR     XDRAW
        ⋮
```

以上程式即相當於BASIC下列指令，但速度較快。

```
10 HGR：POKE 232,0：POKE 233,32：SCALE=2：
   ROT=0：HCOLOR=3
20 XDRAW 1 AT 64,64
```

第二部份
幾個應用上的觀念

第一章　APPLE音效的產生

第二章　如何分割APPLESOFT　BASIC 程式

第三章　低解像圖形表的設計及製作

第四章　高解像圖形表的製作

第一章　APPLE音效的產生

音響效果對遊戲程式有很大的影響，不但可吸引使用者的注意力，更可使整個遊戲程式更加生動、逼眞。讀者可試著想想看在玩小蜜蜂遊戲時若沒有那些嗡嗡聲及爆炸聲，玩起來感覺如何？本節的目的，便是要爲讀者介紹遊戲程式中的聲音是如何產生的。

在APPLE Ⅱ 中，有一個開關位於$C0 30的位址，只要觸動此開關（如 LDA $C0 30），便可使喇叭發出微細的敲擊聲。因此不但能藉著快速的開啓及關閉（觸動）此開關，使喇叭發出聲音，且改變每秒鐘的敲打頻率便可使喇叭發出不同的聲音，製造各種效果。

首先我們看看下面的組合語言程式：

```
030A - AD   30   C0      LDA   $C030
030D - 88                DEY
030E - DO   05           BNE   $0315
0310 - CE   09   03      DEC   $0309
0313 - FO   09           BEQ   $031E
0315 - CA                DEX
0316 - DO   F5           BNE   $030D
0318 - AE   08   03      LDX   $0308
```

```
031B- AC  0A  03        TMP   $030A
031E- 60                RTS
```

程式中利用$0308 中的值決定觸動喇叭的頻率，故能決定音調的高低(PITCH)，若值愈小便能很快的JUMP $030A去觸動喇叭，即每秒頻率較高，而產生較高的音調，相反的若$0308中的值愈大，則產生的聲音便較低。位址$0309則是用以控制聲音的長短(Duration)，其值愈大則聲音愈長，愈小則聲音愈短。由此可知，我們只要將不同的值存入$0308及$0309的位址中，便能產生不同高低、長短的聲音，眞的嗎？讓我們再試試下面的例子：

```
10   CALL 1000 : REM LOAD TONE SUBROUTINE
100  REM SET MUSIC，P=PITCH，D=DURATION
     200 PLAY IT
110  P=255 : D=250 : GOSUB 200 : P=170 : D=250 :
     GOSUB 200 : P=190 : D=75 : GOSUB 200 :
     P = 203 : D=72 : GOSUB 200
120  P=230 : D=75 : GOSUB 200 : P=126 : D=250
     GOSUB 200 : P=170 : D=250 : GOSUB 200 :
     P=190 : D=75 : GOSUB 200
130  P=203 : D=75 : GOSUB 200 : P=230 : D=75 :
     GOSUB 200 : P=126 : D=250 : GOSUB 200 :
     P=170 : D=250 : GOSUB 200
140  P = 190 : D = 100 : GOSUB 200 : P=203 :
     D = 100 : GOSUB 200 : P = 190 : D = 100 :
```

```
     GOSUB 200: P=230: D=250: GOSUB 200:
     END
200  POKE 776,P: POKE 777,D: CALL 778:RET-
     URN
1000 POKE 776,255  : POKE 777,255:
     POKE 778,173  : POKE 779, 48:
     POKE 780,192  : POKE 781,136:
     POKE 782,208  : POKE 783,  5:
     POKE 784,206  : POKE 785,  9: POKE
     786,3: POKE 787,240
1010 POKE 788,9 : POKE 789,202: POKE
     790,208: POKE 791,245: POKE
     792,174: POKE 793, 8: POKE 794
      ,3: POKE 795,76: POKE 796,10
      : POKE 797,3: POKE 798,96: RETURN
```

將此程式鍵入後試著執行一次，果然不錯，這個程式“唱”出了星際大戰（STAR WARS THEME）的音樂。在此程式中第10,1000,1010 行只是將發聲的機器語言程式存入記憶體中，而由110至200行分別用不同的數值（P，D）存入＄0308與＄0309，使APPLEⅡ發出不同音調及長短的聲音。

事實上，目前市面上許多遊戲程式的發聲部份，都是利用上述所講的方式產生的。明白了星際大戰音樂的發音方式後，接著而來的問題便是：如果想讓APPLE唱出別的曲調，則P和D應用什麼值存入＄0308和＄0309的位址中呢？且若長一點的音樂，在程式中豈不要 POKE 半天，有沒有更好的方法呢？讓我們

繼續看下去。

下面再介紹一個稍爲改進的組合語言發聲程式：

```
0320-AC 1F 03    LDY  $031F     ;USE"PITCH"
0323-B9 49 03    LDA  $0349,Y   ;AS OFFSET TQ
0326-8D 1F 03    STA  $031F     ;ACTUAL PITCH
0329-A0 00       LDY  #$00      ; INIT.24BITS
032B-EE 1D 03    INC  $031D     ;COUNTER
032E-EE 1E 03    INC  $031E     ;
0331-AE 1F 03    LDX  $031F     ;RESTORE PITCH
0334-AD 30 C0    LDA  $C030     ;COUNT & CLICK
0337-88          DEY            ;24BITS COUNTER
0338-D0 0A       BNE  $0344     ;COUNTDOWN
033A-CE 1D 03    DEC  $031D
033D-D0 05       BNE  $0344
033F-CE 1E 03    DEC  $031E
0342-FD 05       BEQ  $0349
0344-CA          DEX            ;CHECH NEXT CLICK
0345-F0 EA       BEQ  $0331
0347-D0 EE       BNE  $0337
0349-60          RTS
```

在正式執行此組合語言程式之前，我們還必須將我們的音調數值對應表（PITCH TABLE）存入$034A至$0351的位置中即：

```
]CALL-151
*34A:FF E4 CB CO AB 98 87 7F
*
```

其中$FF 相當於音階中的$D_0$，…，$87則相當於Si，$7F爲

高音的Do。 將此音調數值對應表鍵入後，讓我們回頭看看程式。

程式中利用＄031D及＄031E存放音調的長短（Duration），即利用16個位元存放其數值（＄031F存放音調之高低（PITCH）。在此程式中，代表音調高低的數值爲1至8，相當於音階中的八個音（Do，Re，……，Do），而不是0至255的數值。程式中第1至第3個指令，會自動將此1至8的數值按音調數值對應表換成0至255的數值，以後的過程便類似前面所介紹的發聲程式。

下面我們按照修改過後的發聲組合語言程式，再設計一個BASIC程式，使APPLE按照我們所輸入的簡譜及拍子“唱出”我們所指定的音樂來：

```
10 T$ =“320:AC IF 03 B9 49 03 8D
   1F 03 A0 00 EE 1D 03 EE 1E 03
   AE 1F 03 AD 30 C0 88 D0 0A CE
   1D 03 D0 05 CE 1E 03 F0 05 CA
   F0 EA D0 EE 60 FF E4 CB C0 AB
   98 87 7F”
20 T$=T$ +“ND823G”
30 FOR I = 1 TO LEN(T$)
40 POKE 511 + I,ASC(MID$(T$,I,1))
   + 128
50 NEXT I : POKE 72,0 : CALL -144
60 P$ =“ 88556653586565365353 21 ”
65 D$ =“ 11111211113221122211 2”
70 FOR I=1 TO LEN( P$)
```

```
80 PITCH＝ASC（MID$（P$，I，1））－48
85 DURA＝（ASC（MID$（D$，I，1））－48）*100
   : DURA%＝DURA／256
90 IF PITCH＜1 AND PITCH＞8
   THEN 200
100 POKE 799，PITCH
110 POKE 797，DURA－DURA% * 256：
    POKE 798，DURA%
120 CALL 800
130 NEXT I
200 END
```

鍵入程式後，執行一次，果然不錯，唱出了兒歌“只要我長大”中的一段旋律。程式中第10至50行將前面修改過後的機器語言發聲程式連同8位元組的音調數值對照表存入位址＄320開始的記憶區，其中第20 行目的是使程式回到APPLESOFT的控制。P＄字串表示出我們所想要的音樂的簡譜形式，而D＄則表示P＄中每一個對應位置的音調的相對長短，若想使整首曲子唱得快些，只須將第85 行中的100改成較小的值即可。自第70行至130行，不斷將P$，D$中的字串逐字取出，換成1至8的數值與100的倍數的數值，並存入位址＄031D至＄031F用以控制機器語言發聲程式的音調高低及長短。

讀者只要改變P＄及D＄ 的內含值，便可做出不同的音樂。但僅有8個音，好像“音域太窄了一些”下面便是一個較完整的半音階相對頻率表：

$FF(255)，$F2(242)，$E4(228)，$D7(215)，$CB(203)，$C0(192)，$B5(181)，$AB(171)，$A1(161)，$98(152)，$8F(143)，$87(135)，$7F(127)，$78(120)，$71(113)，$6B(107)，$65(101)，$5F(95)，$5A(90)，$55(85)，$50(80)，$4B(75)，$47(71)，$43(67)，$3F(63)，$3B(59)，$38(56)，$35(53)，$32(50)，$2F(47)，$2C(44)，$2A(42)，$28(40)，$25(37)，$23(35)，$21(33)，$20(32)，$1E(30)，$1C(28)，$1A(26)

在前面所用的音階中的8個音就是自此表的第1，3，5，6，8，10，12，13個位置的8個數值，讀者可類推下去。

第二章　如何分割 APPLESOFT BASIC 程式

APPLE Ⅱ的高解像圖面第一頁與第二頁分別在＄2000 及＄4000，而APPLESOFT BASIC程式由＄800放起，若程式長一點常常會佔用高解像圖面記憶區。此時若程式又使用到高解像圖形，則執行時便會使原BASIC程式被破壞。爲了解決這個問題，以下爲讀者介紹一個程式：SPLITTER。此程式用組合語言寫成目的是將一APPLESOFT BASIC 程式分成兩半儲存，空出兩部份中間的記憶區做其他用途。由於本程式使用到一些APPLESOFT的指位器，因此讀者若對APPLESOFT還不清楚最好先閱讀本書第二章。

SPLITTER（見本章後所附程式）中所使用到的APPLESOFT指位器，與APPLESOFT所使用的名稱並不相同，但對應關係已在程式中第5至10行說明，其中PPL與PPH 即爲TXTTAB；VPH爲 VARTAB之高位元組（即＄6A）AVH 爲ARYTAB之高位元組（即＄6C）；PEL和PEH爲PRGEND。在第12行中，將程式的儲存位置定在＄9400，在DOS 佔用區的正下方（若有DOS），即48K系統（含DOS）的HIMEM的正下方。事實上由第二章我們可知道HIMEM：正下方的記憶區在APPLESOFT中是供儲存字串用的，故記憶區中若存在APPLESOFT BASIC 並使用到字串，則執行此BASIC程式會破壞此SPLITTER程式。故在執行BASIC 程式之前，

必須先將此 SPLITTER 存入磁碟或磁帶。

因SPLITTER的目的是將APPLESOFT 程式分開空出一塊記憶區，故使用者必須在＄6 存入此空隔之大小（以頁數—256位元組—爲單位），並將間隔區起始位置的高位元組存入＄7中（以16進位表示）低位元組設爲＄00。因分開後原BASIC 程式會多空出一塊區域故同時須調整PRGEND（PEH，L）、VARTAB（VP）、ARYTAB（AV）各指位器，將其向高位記憶區移動此間隔外加256個位元組的大小。此外加的256 個位元組的空間，主要是防止將BASIC程式中的某一行從中切斷。（所以選擇256個位元組的原因爲：在輸入時一個程式行不會超過256個字元，故一行BASIC程式也不會超過256 個位元組）。

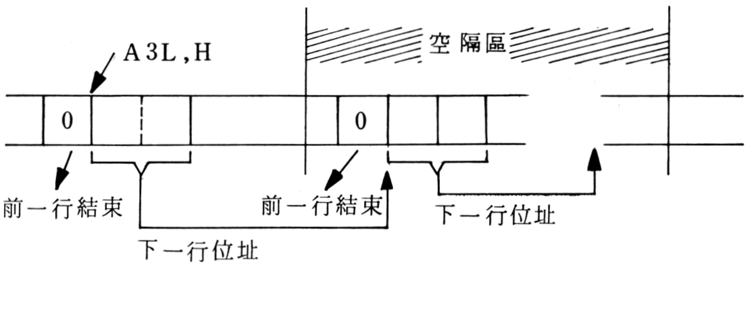
程式中第 19至 27 行，設定開始移動的位址（A3L，A3H），及其目的地的位址（A4L，H），此移動爲由程式最後一個位元組先移動，再移倒數第二個位元組……以此類推。故A3L，H此時內容即爲原程式的結束位址PRGEND之內容，A4L，H之內容爲原程式結束位址加上空隔之大小及額外256個位元組的位址，即（A4L，H）=（PRGEND）+（＄6+1）*256。其中（＄6+1）的動作即由第 20行SEC及 23行ADC＄6 完成。第57至62行即仿照上述方式重新設定程式結束位址與變數開始儲存位址。

程式第30至34行，由程式結束位址向後加上6個零。使APPLESOFT 解譯器執行LIST命令或GOTO、GOSUB 時可辨識程式的結束，此外這些零亦可供SPLITTER辨認結束位址。

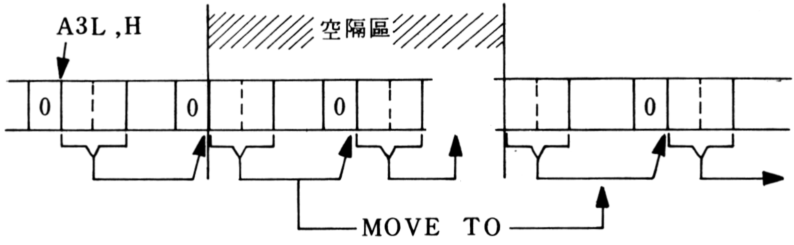
第38至 49行執行移動的動作，Y暫存器此時內含值爲零，

不會產生影響。每當一頁移動後，第46至49行將A3、A4的高位元組減1並檢查是否已到達使用者在$7指定的位址（頁數）的下一頁。讀者注意到第48行CMP $7中與$7比較的A一暫存器中的值爲A3H尚未減1前的值，故當A3H與$7的內含值相同時，除了會將A3H之內含值減1外並會回到MORE再移動256個位元組的程式，即在使用者所指定開始移動頁數位址的前一頁，才是眞正開始移動的位址。移動結束後A3H之內含值已小於$7之內含值，由第54至56行，將原程式眞正被開始移動的位址（頁數）存入$8中（即（$7）-1）。

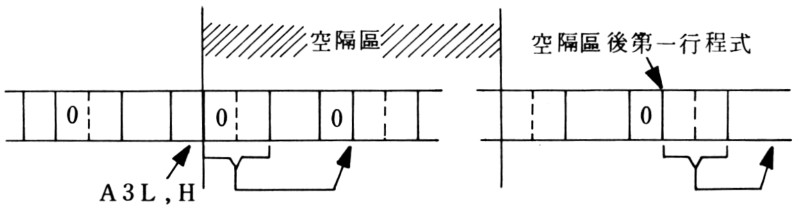
第67至70行更新A3L，H的內含值，使其指向BASIC程式的開頭第一個位元組，此位元組即爲圖2-2中"下一命令行之位址"的低位元組（LDB）。第76、77行檢查下一行是否已進入空隔區若是則跳到PREFIX去執行，若否則由78行至84行由A3L，H所指的BASIC程式行向下一行一行檢查，直到進入空隔區的範圍。如圖所示：



但有一種情形須注意即：A3L，H所在行數之下一行起始位址可能正好是空隔區的起始位址，如下圖所示：

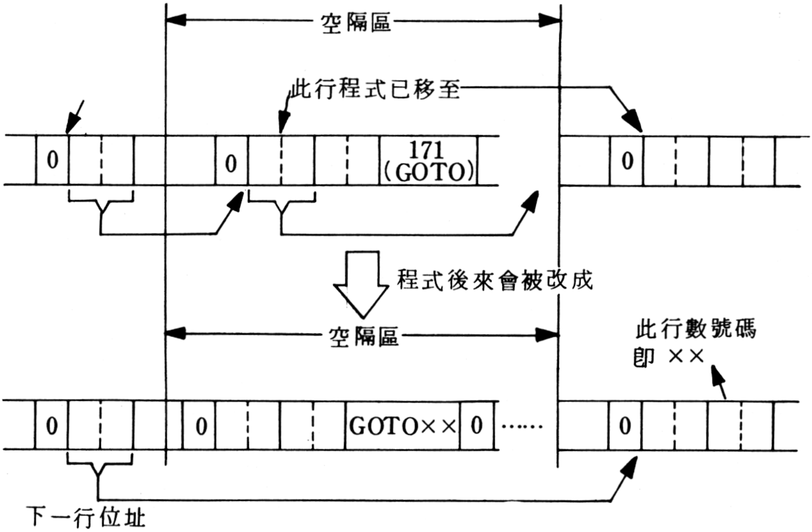


碰到這種情形，可知原來在空隔區的第一行已被移至空隔區後的第一行。但此時產生一個問題，即空隔區後的第一行開始位址之前沒有前導位元 0 代表前一行的結束，故必須再向下前進一行，使A3L,H 指向空隔區的起始位址（如下圖）。程式中第89 至 95 行即負責此項工作。



程式進行至此A3L,H 指著空隔區前的最後一行BASIC 程式，第100至 103 行增加Y暫存器之值，使自 104 至 105 行的檢查此行（A3L，H所指之行）結束位元組＄00 的過程跳過二位元組的“下一行位址”的指標及二位元組的行數號碼。第 106 至 110 行，找到下一行存指令代碼的位置（此行爲A3L,H所指行數的下一行，事實上已移至空隔區之後），並在 111、112 行

將GOTO的指令代碼171存入，造出一個GOTO的命令句。在往後的步驟中，並將空隔區後第一行的行數號碼以ASCⅡ碼的形式存入GOTO之後，見下圖：



在BASIC程式被順序執行時並不根據“下一行位址”的指標找到下一行再執行，故我們加上一GOTO×××××的命令，使BASIC解譯器能執行“眞正”的下一行；且讀者須注意的是，在LIST時並不會將此行（GOTO ×××××）印出，因我們已將GOTO×××××前一行的“下一行位址”指標改成指到“眞正”的下一行，同理對原來程式中的GOSUB及GOTO也不會有影響。

第113至116行將A3L，H之值存入A4L，H，並將Y值存入$9，使我們在稍後的步驟中能很容易的找到GOTO指令碼

的下一個位元組的位址。第 119 至 123行求出空隔區後第一行的位址並將其存入A3L，H 所指行數的“下一行位址”指標的高位元組中（因空隔是以 256個位元組爲單位，低位元組內容相同）。第 124 至 128 行使A3L,H指在空隔區後第一行的起始位址，129 至 131 行使程式跳至REFIX段去執行（因＄9之內含值絕不會是零），且因Y暫存器此時內含值爲 1，至 151行的INC使Y暫存器的內含值成爲 2，故在第 152行能取出此重新定位行（relocated line)的第 3 個位元組，即存放行數號碼的低位元組。＄19及＄1A用來存放此行數號碼之值（BINNO），153至 156 行完成此項儲存動作。

此時必須將此二進位的行數號碼改成ASCII 碼存入先前未完成的GOTO命令行中。在 146 行的DCTAB 的 5 個數值分別對應於十進位的 10000、1000、100、10、1，但將高低位元組次序相反如對應於 10000 的＄2710 存爲＄1027（因6502中低位元組在先高位元組在後）。第 157 至 160 行使DECPTR指標指在 10000之值上，並利用X暫存器內含的正負值代表行數號碼小於（X > 0）或大於（X < 0）32767，使大於32767 的行數號碼不會被視爲負數。163 至 165 行將ASC(“O”)－1 之值存入BLDDEC中配合 165行的INC BLDDEC。以下的步驟爲：將BINNO所代表的行數號碼不斷的減 10000，直到出現負值，而求出行數號碼以十進位表示的萬位數，再將負值加上10000成爲小於 10000之值，再減去 1000，…… 重複以上步驟便可將其改成十進位的五位數。第 166 至 176行自BINNO所存的行數號碼二進位值中減去 10的某次方（視 10000、1000、……而定），若相減後仍爲正值則再減一次，若負值則由 178 至179行檢查此負值是因大於 32767而成爲負值或是眞正由正值減成負

值（由X暫存器之正負可判斷）。若原來行數號碼值大於32767，X暫存器內容小於0，但經不斷的減去 10000 後此行數號碼值終必小於32767，此時在174 行會將X暫存器之內容改爲正值。一旦BINNO所代表的行數號碼值成爲眞正的負值，181至188行會將10 的某次方重新加回，並開始求下一位的十進位數字。在這部份程式中，Y暫存器之值指示目前減數爲DCTBL的第幾個數值，並在189行中將其值存入位址＄1E，而將Y暫存器暫時移作他用。在116和118 行中我們將存此ASCⅡ行數號碼的位址存入A4L，H及＄9中，而由190至192行將求出的ASCⅡ行數號碼的數字存入，並由193至195 行在此存入數字的下一個位元組存入＄00。當然若此時存入的數字爲萬、千、百、拾位則此時在其後再存入的＄00 會被下次存入的千、百、拾、個位所取代，但若此時存入的數字若爲個位數則其後的＄00便不會被取代而成爲BASIC程式中每行結束的信號（ie GOTO ×××××此命令行結束的信號）。

第203與205行將＄9位址及Y暫存器的內容分別設定爲0與1，使得205行能將程式轉移至AFIX及FIX段去執行。讀者可能已注意到程式執行至此，移至空隔區後的各程式行中的“下一行位址”仍爲未移動前的位址，故在此時的FIX 段中，由A3L，H一行接一下的將此“下一行位址”指標的高位元組更改，直到A3L，H到達程式的盡頭（此時A3L，H 所指位址的內容即先前我們在30至34行所事先設定的零）。最後132及133行使程式執行139行，JUMP至＄0 而回到APPLESOFT控制之下。

到此已將此 SPLITTER 程式說明完畢，讀者可發現並不難，只是反覆的運用幾個指標。最後作者希望，除了利用這篇程

式之外，每一位讀者更能跟著本文將程式從頭到尾仔細閱讀一遍，相信必定獲益匪淺。

```
SOURCE FILE: SPLITTER
0000:            1 ********************************
0000:            2 **     APPLESOFT  SPLITTER     **
0000:            3 **              BY             **
0000:            4 **     CHARLES  KLUEPFEL       **
0000:            5 **                             **
0000:            6 ********************************
0040:            7 A3L       EQU   $40
0041:            8 A3H       EQU   $41
0042:            9 A4L       EQU   $42
0043:           10 A4H       EQU   $43
0067:           11 PPL       EQU   $67
0068:           12 PPH       EQU   $68
006A:           13 VPH       EQU   $6A
006C:           14 AVH       EQU   $6C
00AF:           15 PEL       EQU   $AF
00B0:           16 PEH       EQU   $B0
0000:           17 *
0000:           18 *
----- NEXT OBJECT FILE NAME IS SPLITTER.OBJ0
9400:           19           ORG   $9400
9400:38         20           SEC
9401:A5 B0      21           LDA   PEH
9403:85 41      22           STA   A3H
9405:65 06      23           ADC   $6
9407:85 43      24           STA   A4H
9409:A5 AF      25           LDA   PEL
940B:85 40      26           STA   A3L
940D:85 42      27           STA   A4L
940F:           28 * INDICATE END OF PROGRAM TO FIX
940F:           29 *
940F:A0 06      30           LDY   #$6
9411:A9 00      31 RET       LDA   #$0
9413:91 42      32           STA   (A4L),Y
9415:88         33           DEY
9416:D0 F9      34           BNE   RET
9418:           35 *
9418:           36 * MOVE DONE HERE
9418:           37 *
9418:B1 40      38 MORE      LDA   (A3L),Y
```

```
941A:C6 40              DEC  A3L
941C:91 42              STA  (A4L),Y
941E:C6 42              DEC  A4L
9420:A5 42              LDA  A4L
9422:C9 FF              CMP  #$FF
9424:D0 F2              BNE  MORE
9426:A5 41              LDA  A3H
9428:C6 43              DEC  A4H
942A:C6 41              DEC  A3H
942C:C5 07              CMP  $7
942E:10 E8              BPL  MORE
9430:          *
9430:          * NOW RESET PROGRAM END
9430:          * POINTERS & VARIABLE PTRS
9430:          *
9430:A5 07              LDA  $7
9432:85 08              STA  $8
9434:C6 08              DEC  $8
9436:38                 SEC
9437:A5 B0              LDA  PEH
9439:65 06              ADC  $6
943B:85 B0              STA  PEH
943D:85 6A              STA  VPH
943F:85 6C              STA  AVH
9441:          *
9441:          * NOW SET A3L TO BEGIN OF
9441:          * PROGRAM
9441:          *
9441:A5 67              LDA  PPL
9443:85 40              STA  A3L
9445:A5 68              LDA  PPH
9447:85 41              STA  A3H
9449:          *
9449:          * FIND START OF CHANGED AREA
9449:          *
9449:A0 01              LDY  #$1
944B:B1 40     CONT     LDA  (A3L),Y
944D:C5 08              CMP  $8
944F:F0 0B              BEQ  PREFIX
9451:AA        CONT2    TAX
9452:88                 DEY
9453:B1 40              LDA  (A3L),Y
9455:85 40              STA  A3L
9457:86 41              STX  A3H
```

```
9459:C8                    INY
945A:D0 EF                 BNE   CONT
945C:            *
945C:            * CHECK TO BE SURE NOT FIRST
945C:            * BYTE OF PAGE
945C:            *
945C:88          PREFIX    DEY
945D:B1 40                 LDA   (A3L),Y
945F:F0 02                 BEQ   CONT1
9461:D0 05                 BNE   FIX1
9463:C8          CONT1     INY
9464:B1 40                 LDA   (A3L),Y
9466:D0 E9                 BNE   CONT2
9468:            *
9468:            * FIND NEXT LINE AND MAKE IT
9468:            * A GOTO
9468:            *
9468:C8          FIX1      INY
9469:C8                    INY
946A:C8                    INY
946B:C8          FIX2      INY
946C:B1 40                 LDA   (A3L),Y
946E:D0 FB                 BNE   FIX2
9470:C8                    INY
9471:C8                    INY
9472:C8                    INY
9473:C8                    INY
9474:C8                    INY
9475:A9 AB                 LDA   #171         ;GOTO
9477:91 40                 STA   (A3L),Y
9479:A5 40                 LDA   A3L
947B:85 42                 STA   A4L
947D:A5 41                 LDA   A3H
947F:85 43                 STA   A4H
9481:C8                    INY
9482:84 09                 STY   $9
9484:A0 01                 LDY   #1
9486:B1 40                 LDA   (A3L),Y
9488:38          FIX       SEC
9489:65 06                 ADC   $6
948B:91 40                 STA   (A3L),Y
948D:88                    DEY
948E:AA                    TAX
948F:B1 40                 LDA   (A3L),Y
```

```
9491:85 40      127             STA   A3L
9493:86 41      128             STX   A3H
9495:C8         129             INY
9496:A6 09      130             LDX   $9
9498:D0 11      131             BNE   REFIX
949A:B1 40      132 AFIX        LDA   (A3L),Y
949C:D0 EA      133             BNE   FIX
949E:           134 *
949E:           135 * WHEN YOU REACH THOSE '0'S
949E:           136 * AT END , FALL THROUGH
949E:           137 *
0000:           138 OUT         EQU   $0
949E:4C 00 00   139 STOP        JMP   OUT
94A1:           140 *
94A1:           141 * SET UP GOTO LINE NO.
94A1:           142 *
0019:           143 BINNO       EQU   $19
001B:           144 DECPTR      EQU   $1B
001D:           145 BLDDEC      EQU   $1D
94A1:10 27      146 DCTBL       DDB   $1027
94A3:E8 03      147             DDB   $E803
94A5:64 00      148             DDB   $6400
94A7:0A 00      149             DDB   $0A00
94A9:01 00      150             DDB   $0100
94AB:C8         151 REFIX       INY
94AC:B1 40      152             LDA   (A3L),Y
94AE:85 19      153             STA   BINNO
94B0:C8         154             INY
94B1:B1 40      155             LDA   (A3L),Y
94B3:85 1A      156             STA   BINNO+1
94B5:A9 A1      157             LDA   #>DCTBL
94B7:85 1B      158             STA   DECPTR
94B9:A9 94      159             LDA   #<DCTBL
94BB:85 1C      160             STA   DECPTR+1
94BD:A0 01      161             LDY   #1
94BF:A2 01      162             LDX   #1
94C1:A9 2F      163 NEWSUB      LDA   #$2F
94C3:85 1D      164             STA   BLDDEC
94C5:E6 1D      165 SUBT        INC   BLDDEC
94C7:A5 19      166             LDA   BINNO
94C9:88         167             DEY
94CA:38         168             SEC
94CB:F1 1B      169             SBC   (DECPTR),Y
94CD:85 19      170             STA   BINNO
```

```
94CF:C8                 INY
94D0:A5 1A              LDA   BINNO+1
94D2:30 02              BMI   NG
94D4:A2 00              LDX   #$0
94D6:F1 1B         NG   SBC   (DECPTR),Y
94D8:85 1A              STA   BINNO+1
94DA:10 E9              BPL   SUBT
94DC:8A                 TXA
94DD:D0 E6              BNE   SUBT
94DF:A5 19              LDA   BINNO
94E1:88                 DEY
94E2:18                 CLC
94E3:71 1B              ADC   (DECPTR),Y
94E5:85 19              STA   BINNO
94E7:C8                 INY
94E8:A5 1A              LDA   BINNO+1
94EA:71 1B              ADC   (DECPTR),Y
94EC:85 1A              STA   BINNO+1
94EE:84 1E              STY   $1E
94F0:A5 1D              LDA   BLDDEC
94F2:A4 09              LDY   $9
94F4:91 42              STA   (A4L),Y
94F6:C8                 INY
94F7:A9 00              LDA   #0
94F9:91 42              STA   (A4L),Y
94FB:E6 09              INC   $9
94FD:A4 1E              LDY   $1E
94FF:C8                 INY
9500:C8                 INY
9501:98                 TYA
9502:C9 0A              CMP   #10
9504:30 BB              BMI   NEWSUB
9506:86 09              STX   $9
9508:A0 01              LDY   #1
950A:4C 9A 94           JMP   AFIX

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

第三章　低解像圖形表的設計及製作

精彩的圖形顯示是APPLE Ⅱ 微電腦的一項特徵，尤其是高解像圖形表的各項畫圖功能，更使得在高解像螢幕上的圖形顯示與圖形的控制簡單而又靈活（但圖形表的建立仍是一件繁瑣的工作，在下一章會介紹一種利用軟體自動建立高解像圖形表的方法）。但在低解像畫面上，並沒有類似高解像畫面圖形表及 'DRAW'，XDRAW' 的指令，而必須利用 'PLOT'，'HLIN'，'VLIN' 等一些畫圖指令畫出低解像圖形，利用這種方式畫圖的缺點是繪圖不易及速度不夠快。因此本章的目的便是要為讀者介紹一種低解像圖形表的方法，使我們能模仿高解像圖形表很輕易的畫出複雜的低解像圖形並加以消除或移動。

在後面的海空大戰（AIRSEA BATTLE）的遊戲中，便是利用此低解像圖形表的方法，控制戰鬥機、轟炸機、潛艇、炸爆…等一共 7 個圖形的顯示或移動。讀者將可由此程式中感覺到這種繪圖方式的二大優點：簡單、速度快。

介紹

本文一共包括二個程式，一個利用APPLESOFT BASIC 所寫成，我們可稱其為低解像圖形表製作程式（SHAPE MAKER），用以建立低解像圖形的圖形表；第二個程式則由迷你組合語言寫成，可稱為低解像圖形顯示程式（SHAPE WRITER）

，用以將圖形表所代表的圖形繪出，爲求快速的畫出圖形，故這部份程式由迷你組合語言所撰寫。

執行程式時，必須考慮到記憶區分配問題。在此程式中，將顯示圖形的機器語言置於＄300－＄371 第三頁記憶區中，並設定HIMEM：38450（＄9600）—— 假設沒有磁碟作業系統及48K機器——使低解像圖形表可存入＄9600以上的記憶區。當然這種記憶區分配方式並不是絕對的，讀者可根據自己的應用，選擇其他方式。例如也可以不更動HIMEM：而將圖形表存入高解像第二頁記憶區中。

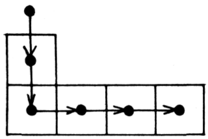
程式一：低解像圖形表製作程式（LOW-RES SHAPE MAKER）。

此程式是模仿高解像圖形表的方式建立低解像圖形表，其圖形表表示方式如下：

動 作	畫 點	不 畫 點
上	10 ($A)	1 ($1)
下	11 ($B)	2 ($2)
右	12 ($C)	3 ($3)
左	13 ($D)	4 ($4)

（＊） ＄00代表圖形表結束

如：



圖形可表示爲：

OB OB OC OC OC OO

由於組成低解像圖形的點數並不會很多，且爲了加快圖形的顯示速度，程式中以一個位元組存一個畫點動作（不像高解像圖形表一個位元組存入2至3個畫點動作）。並以按鍵 'U'（UP），'D'（DOWN），'R'（RIGHT），'L'（LEFT），分別代表上、下、左、右的移動；'P'（PLOT）代表是否畫出點來或是

僅移動而已；'F'（FINISH）鍵用以結束圖形表。

程式一開始顯示說明，並請使用者輸入圖形表欲存入的起始位址（程式中建議的位址爲＄9700）及選定螢幕上的一個位址爲圖形顯示的起始位置。隨後便按使用者輸入的鍵在螢幕上創作圖形，並將各畫點動作代碼存入使用者所指定的位址中。至使用者按下'F'鍵後，程式自動在圖形表最後補上＄00代表圖形表已完成。並可依使用者的要求在畫面上移動，以測試效果。此外程式並會將圖形表的起始位址及圖形表長度印出，供使用者存入磁碟或磁帶中。

本程式可使使用者在螢幕上很容易的描繪低解像圖形，並在螢幕上顯示，自動依圖形表規則存入指定圖形表位址，並告知起始位址及長度，使用者在圖形製作完畢後，可將本程式消除，存入眞正使用圖形表的主程式，如後面的「海空大戰」遊戲程式。

此程式雖是模擬高解像圖形表顯示方式，但基本上有一個很大的差別即：高解像圖形表的顯示方式是先決定畫點不畫點之後，再移至指定的位置，故對於"上移並畫點"的動作，是先在原位置畫出一點後再移至新位置；但在這裡採用的方式爲先移動至新位置後再在新位置畫出一點（關於這點，讀者可在圖形表顯示程式中很明顯的看出）。

以下我們對此圖形表製作程式的流程及變數做一個說明。

主要變數：

(1)PT：PLOT FLAG，大於0時代表目前是屬於移動並畫點的狀態，小於0時代表只是移動而不畫點。

(2)PPT：PREVIOUS PLOT FLAG，指示前一個畫點動作是屬於PLOT ON或PLOT OFF的狀態。因程式在螢幕上控制一個游標，使它在目前圖形進行到的位置閃爍，使使

用者知道目前進行的位置，但卻使此位置是否有畫出無法判斷，因此程式除了在螢幕底端說明此位置是PLOT ON或OFF之外，並會在游標移至下一個位置之後，便根據此PPT之值畫出（或不畫出）此點。

(3)XD，YD ：目前進行到的點之螢幕座標位置。

(4)XE，YE ：前一個點的螢幕座標位置。

(5)ST：圖形表的起始儲存位址。

(6)LOC：每一個畫點動作代碼的儲存位址，初值即為ST。

程式說明：

(1)10-30行：顯示程式說明，並請使用者輸入圖形表之起始儲存位址及螢幕上圖形開始描繪的位置。

(2)33-35行：設定低解像圖形文字混合幕，並在螢幕下方指示描繪圖形的方法（35行）。

(3)510-595行：在目前圖形描繪點所在位置顯示游標，並隨時檢查鍵盤上是否有鍵被按下。若有則至各副程式分別處理。

(4)600-659行：600 行檢查圖形是否已超過螢幕範圍，若已超過則至1400 行的副程式顯示警告訊息。605至659 行會在原游標顯示點依此點應畫出或不畫出（由PPT決定）予以適當的顯示後，並在螢幕底下以文字說明此時為“PLOT ON或PLOT OFF”及新的游標所在位置之點是屬於“PLOT ON”或PLOT OFF” 兩種訊息。最後回到510 行顯示最新游標位置（即圖形的最新描繪位置）並等候使用者的輸入資料。

(5)700-760行：為一副程式，當使用者按下“F”鍵時，便跳到此副程式執行。首先在圖形表最後補上兩個零，做為圖形表的結束，並顯示圖形表的儲存位址及其長度。此外並詢問使

用者是否想測試剛才所描繪的圖形（測試部份即由708至750行進行圖形的顯示及移動）。最後在755行詢問讀者是否繼續描繪新的圖形或終止程式之執行。

(6) 1000-1350：包含四個副程式，分別處理上、下、左、右動作所對應的圖形代碼，並存入圖形表中。

(7) 1400-1450 行：當使用者在螢幕上描繪的圖形超出螢幕的範圍時，便由此副程式顯示警告訊號。碰到這種情形，對圖形表本身並沒有錯誤產生，只是提醒使用者螢幕上所顯示的圖形和圖形表所代表的圖形已有差異。

本程式最主要是介紹低解像圖形表的概念，讀者還可以對程式配合自己的需要加以修改。例如有磁碟機的讀者可加上自動將產生的圖形表存入磁碟的能力。此外本程式只能產生圖形表，卻沒有修改圖形表的能力，讀者何不動手對本程式再加以擴展，增加對一已存在的圖形表修改某一部份或某一點的能力，使本程式由“SHAPE MAKER”成為“SHAPE EDITOR”。

```
]LIST

10  HIMEM: 38405:PT = 1
20  HOME : VTAB 5: HTAB 4: INVERSE
     : PRINT "** LOW RESOLUTION S
     HAPE MAKER **": VTAB 6: HTAB
     8: PRINT "<< VERSION 1.0 >>"
     : NORMAL
21  PRINT : PRINT "THIS PROGRAM A
     LLOWS YOU TO USE THE KEY-": PRINT
     "BOARD TO CREATE LOW-RES GRA
     PHIC SHAPES."
22  PRINT : PRINT "U=UP D=DOWN R=
     RIGHT L=LEFT F=FINSH"
24  PRINT : PRINT "YOU CAN TURN P
     LOT ON/OFF BY PRESSING P."
25  PRINT : PRINT "AFTER ENTERING
      THE SHAPE, THE PROGRAM": PRINT
      "GIVES THE ADDRESS OF THE SHAPE"
```

```
27  PRINT "YOU CAN USE IN YOUR OWN PROG
     RAM"
29  PRINT : PRINT "ENTER STARTING ADDRE
     SS IN DECIMAL": PRINT "(RECOMMENDE
     D: 38656 STARTS THE SHAPE": INPUT
     "AT HEX $9700) :";LOC:ST = LOC
30  HOME : VTAB 10: PRINT "WHERE IS THE
      STARTING POINT OF SHAPE?": INPUT
     "X= ";XD: INPUT "Y= ";YD
33 XE = XD:YE = YD:PPT =  - 1: GR
34  CALL  - 936: VTAB 21: PRINT "U=UP D
     =DOWN R=RIGHT L=LEFT F=FINSH"
510  COLOR= 3: PLOT XD,YD: COLOR= 0: PLOT
     XD,YD
514 K =  PEEK ( - 16384): IF K < 128 THEN
     510
515 K = K - 128
518  POKE  - 16368,0
520  IF K =  ASC ("U") THEN  GOSUB 1000
530  IF K =  ASC ("D") THEN  GOSUB 1100
540  IF K =  ASC ("R") THEN  GOSUB 1200
550  IF K =  ASC ("L") THEN  GOSUB 1300
560  IF K =  ASC ("P") THEN PT =  - PT: GOTO
     650
595  IF K =  ASC ("F") THEN 700
600  IF XD < 0 THEN  GOSUB 1400:XD = 0
601  IF XD > 39 THEN  GOSUB 1400:XD <  = 39
602  IF YD < 0 THEN  GOSUB 1400:YD = 0
603  IF YD > 39 THEN  GOSUB 1400:YD = 39
605  IF PPT > 0 THEN  COLOR= 6: PLOT XE
     ,YE
607  VTAB 23: IF PT > 0 THEN  PRINT "FL
     ASHING DOT IS ON "
609  IF PT < 0 THEN  PRINT "FLASHING DO
     T IS OFF"
610 XE = XD:YE = YD:PPT = PT
650  VTAB 22: IF PT > 0 THEN  PRINT "PL
     OT ON "
655  IF PT < 0 THEN  PRINT "PLOT OFF"
657  GOTO 510
700  IF PPT > 0 THEN  COLOR= 6: PLOT XD
     ,YD
701  POKE LOC,0:LOC = LOC + 1: POKE LOC
     ,0: TEXT
702  HOME : PRINT "SHAPE COMPLETE
     D. STORE AT LOC: ";ST;"; LEN
```

```
       G: ";LOC - ST
703    PRINT : INPUT "DO YOU WANT T
       O DISPLAY/SHIFT THE SHAPE?(Y
       /N)";A$: IF A$ = "N" THEN 755
704    TEXT : HOME : VTAB 5: PRINT
       "BE SURE YOU SHAPE WITHIN A
       10*10": PRINT "GRID,  GOING
       OUTSIDE THE GRID WILL"
705    PRINT "CAUSE AN ERROR IN ATT
       EMPTING TO SCAN": PRINT "THE SHAPE "
706    INPUT "READ?(Y/N)";A$: IF A$
        = "N" THEN  GOTO 703
707    GR
708    FOR Y = 15 TO 25 STEP 5:H =
        INT (ST / 256):L = ST - H *
       256: POKE 771,L: POKE 772,H
710    FOR X = 12 TO 25: POKE 8,0: POKE
       5,XOLD: POKE 6,YOLD: CALL 768
730    POKE 5,X: POKE 6,Y: POKE 8,1
       19: CALL 768:XOLD = X:YOLD = Y
750    NEXT X,Y
755    VTAB 24: INPUT "ENTER: 1=(NE
       W SHAPE)  2=(END)";E: IF E =
       1 THEN  TEXT : HOME : GOTO 29
760    TEXT : HOME : END
1000    IF PT > 0 THEN  POKE LOC,10
       :LOC = LOC + 1:YD = YD - 1: RETURN
1050    POKE LOC,1:LOC = LOC + 1:YD
        = YD - 1: RETURN
1100    IF PT > 0 THEN  POKE LOC,11
       :LOC = LOC + 1:YD = YD + 1: RETURN
1150    POKE LOC,2:LOC = LOC + 1:YD
        = YD + 1: RETURN
1200    IF PT > 0 THEN  POKE LOC,12
       :LOC = LOC + 1:XD = XD + 1: RETURN
1250    POKE LOC,3:LOC = LOC + 1:XD
        = XD + 1: RETURN
1300    IF PT > 0 THEN  POKE LOC,13
       :LOC = LOC + 1:XD = XD - 1: RETURN
1350    POKE LOC,4:LOC = LOC + 1:XD
        = XD - 1: RETURN
1400    HOME : FLASH : PRINT "OUT O
       F RANGE : X=";XD;"; Y=";YD;
1450    NORMAL : FOR Z = 0 TO 2000:
        NEXT Z: HOME : RETURN
```

程式二：低解像圖形表顯示程式

本程式由組合語言寫成，可將程式一所產生的低解像圖形表顯示在螢幕上。因此，在使用低解像圖形表時，讀者必須在主程式之外加上此程式當作副程式，在需要顯示低解像圖形表時便呼叫此程式（如程式一 710 及 730 行的 CALL 768）。

在呼叫本程式之前須將圖形中央位置（即開始描繪點）的（X，Y）座標分別存入 $ 05 及 $ 06 的位址中。圖形欲顯示的顏色代碼存入 $ 08 中。顏色代碼如下表：

COLOR	$ 08 內含值	COLOR	$ 08 內含值
0	$ 00	8	$ 88
1	$ 11	9	$ 99
2	$ 22	10	$ AA
3	$ 33	11	$ BB
4	$ 44	12	$ CC
5	$ 55	13	$ DD
6	$ 66	14	$ EE
7	$ 77	15	$ FF

使用上述顏色代碼的原因已在第一部份第三章說明，故不贅述。

此程式是以（$ 303，$ 304）所指的位址爲圖形表的起始位址，在後面的程式中爲 $ 9700 是配合程式一使用。程式中 $ 306 至 $ 32 A，用以判斷上、下、左、右的畫點動作，接著利用 $ 05 及 $ 06 分別存放圖形位置的（X，Y）座標值。$ 32 E 至 $ 355 以此座標做上、下、左、右的畫點動作。眞正畫點則是由 PLOT（$ F 800，見第三章）的監督程式副程式完成。若由圖形表所

得的資料是＄01，＄02，＄03，＄04，則由＄356至＄360設定顏色爲黑色（＄30爲＄F800副程式所使用的顏色指示位址）然後才畫黑點；若爲＄OA，＄OB，＄OC，＄D則按使用者存入＄08的顏色值來畫出有顏色的點。

總上所述，利用BASIC來使用此程式應包括下列四個步驟:

1. 將圖形表儲存位址存入＄303，＄304（低、高位元組次序）。
2. 將圖形在螢幕上的顯示位置存入＄05、＄06中。
3. 將圖形的顏色碼存入＄08中。
4. CALL 768。

```
*300LLL

0300-   A2 00       LDX   #$00
0302-   BD 00 97    LDA   $9700,X
0305-   E8          INX
0306-   C9 01       CMP   #$01
0308-   F0 24       BEQ   $032E
030A-   C9 02       CMP   #$02
030C-   F0 25       BEQ   $0333
030E-   C9 03       CMP   #$03
0310-   F0 26       BEQ   $0338
0312-   C9 04       CMP   #$04
0314-   F0 27       BEQ   $033D
0316-   C9 0A       CMP   #$0A
0318-   F0 28       BEQ   $0342
031A-   C9 0B       CMP   #$0B
031C-   F0 29       BEQ   $0347
031E-   C9 0C       CMP   #$0C
0320-   F0 2A       BEQ   $034C
0322-   C9 0D       CMP   #$0D
0324-   F0 2B       BEQ   $0351
0326-   C9 00       CMP   #$00
0328-   F0 03       BEQ   $032D
032A-   4C 02 03    JMP   $0302
032D-   60          RTS
032E-   C6 06       DEC   $06
0330-   4C 56 03    JMP   $0356
```

```
0333-   E6 06       INC   $06
0335-   4C 56 03    JMP   $0356
0338-   E6 05       INC   $05
033A-   4C 56 03    JMP   $0356
033D-   C6 05       DEC   $05
033F-   4C 56 03    JMP   $0356
0342-   C6 06       DEC   $06
0344-   4C 64 03    JMP   $0364
0347-   E6 06       INC   $06
0349-   4C 64 03    JMP   $0364
034C-   E6 05       INC   $05
034E-   4C 64 03    JMP   $0364
0351-   C6 05       DEC   $05
0353-   4C 64 03    JMP   $0364
0356-   A9 00       LDA   #$00
0358-   85 30       STA   $30
035A-   A4 05       LDY   $05
035C-   A5 06       LDA   $06
035E-   20 00 F8    JSR   $F800
0361-   4C 02 03    JMP   $0302
0364-   A5 08       LDA   $08
0366-   85 30       STA   $30
0368-   A4 05       LDY   $05
036A-   A5 06       LDA   $06
036C-   20 00 F8    JSR   $F800
036F-   4C 02 03    JMP   $0302
0372-   FF          ???
0373-   04          ???
0374-   05 06       ORA   $06
0376-   FF          ???
0377-   FF          ???
0378-   FF          ???
0379-   FF          ???
037A-   FF          ???
037B-   FF          ???
```

第四章　高解像圖形表的製作

一個遊戲程式，沒有圖形的陪襯，在效果上便打了很大的折扣，因此圖形的設計便成為我們在設計程式時的一個重要考慮因素。在 APPLE Ⅱ 中APPLESOFT提供我們兩種類形的繪圖方式；一種為利用 HLIN，VLIN，HPLOT ……等畫直線或畫點的方式，這種方式使用起來很方便，但對於不具對稱性較為複雜的圖形，便令人覺得速度太慢且不敷使用，因此第二種利用圖形表的方式，便成為大家所採用的方式。

利用圖形表繪出高解像圖形，雖然圖形較細緻美觀，但卻有一個嚴重的缺點：即圖形表不容易製作。往往我們在紙上畫了半天再加以轉換成16進位值後，顯示的效果並不如我們所預期的。因此我們感覺迫切需要一種“圖形表製作程式”來幫助我們解決此一困難。本章即針對此問題，設計一個簡單的高解像圖形表製作程式，使讀者了解在設計上的基本觀念，進而能夠設計一個較適合本身使用的圖形表製作程式。

程式的使用及基本觀念

本程式的功用與前一章的低解像圖形表製作程式很相似，只是對像不同，一個是用來製作低解像圖形表，而另一個則用來製作高解像圖形表。在閱讀以下程式前，讀者應已對第四章所述的

高解像圖形表的產生原理徹底了解。

程式執行前，使用者須輸入此圖形表應包含幾個圖形及圖形表的起始儲存位址。使程式能按照使用者的意思產生圖形表並存入指定的位址。程式在剛開始會在螢幕上顯示一說明，告訴讀者如何使用此程式製作圖形表：

1. “SPACE BAR”空白鍵：程式執行中的任何時機，使用者均可按下此鍵以顯示使用說明。
2. “U”，“D”，“L”，“R”：在製作圖形表時，程式會在高解像螢幕上顯示一個閃爍的游標點，以指出目前進行到的位置。使用者可利用此四鍵，並配合畫點或不畫點，以移動游標向指定的方向（上、下、左、右）做畫點或不畫點後的移動。
3. “T”：通常在製作圖形表的高解像螢幕是圖形文字混合幕，即在高解像螢幕上保留下面四行做爲顯示文字訊息之用，但使用者可按下此鍵，將此圖形文字混合螢幕轉變爲全部供圖形顯示之螢幕。（MIXED／FULL SCREEN）。
4. “Q”：QUIT，當圖形表中的任一個圖形製作完畢後（一圖形表中可能包含數個圖形），均須按下此鍵代表一個圖形的結束。
5. “E”：按下此鍵後，便跳出此程式之控制，程式將控制權交回 APPLESOFT 編譯程式。
6. “P”：PEN，此鍵控制畫點或不畫點的動作。按下此鍵後可由原來的畫點改成不畫點，或原來的不畫點改成畫點。利用此鍵與“U”，“D”，“L”，“R”，四鍵可指出畫點後的上、下、左、右移動，或不畫點的上、下、左、右移動。

主要變數說明

程式中使用到下列幾個主要變數

(1)BADD： BASE ADDRESS，此變數儲存使用者所輸入的圖形表起始儲存位址。

(2)NUM：此變數儲存使用者圖形表中，應包括的圖形個數。

(3) CNUM：此變數用來儲存除了目前正在製作的圖形外，還有幾個圖形須製作。

(4) PS ：此變數內含值代表程式執行當中爲畫點或不畫點的狀態；0 代表不畫點，1 代表畫點。程式一開始PS 之初值設定爲0，往後使用者按下“ P ”鍵後，便可將PS 之值由0轉變成1（或由1轉變成0）。

(5) ADD：ADDRESS ，代表每一個新產生的圖形代碼的儲存位址。

(6)（X ，Y ）：指示游標點在高解像螢幕上的顯示座標位置。

(7) DIS：DISTANCE，此變數用來記錄圖形表中每一個即將製作的圖形的開始儲存位址與整個圖形表的起始儲存位址之差，以便能存入圖形表中（詳細說明請見第四章，有關圖形表的組成。

(8)TA ：在圖形表的結構上，頭兩個位元組，儲存圖形的個數。假設有N個圖形，則以下的2N個位址，分別儲存各個圖形的開始儲存位址與圖形表起始儲存位址的差距（OFFSET）。而變數TA即用以指出，目前製作的圖形，此差距的儲存位址。我們可用下圖表示：

(9) ST ：此變數用以指示目

前螢幕上爲全部供圖形顯示（FULL SCREEN），或爲文字圖形混合幕（MIXED SCREEN）。ST ＝1代表全螢幕，ST ＝0代表混合幕。

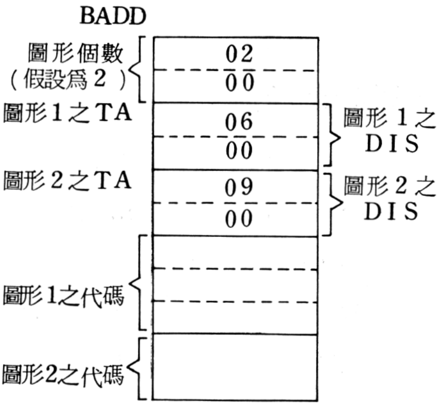


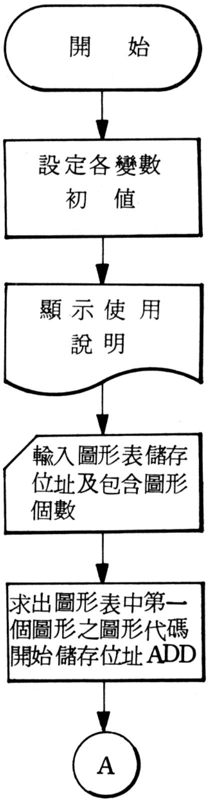
圖 形 表

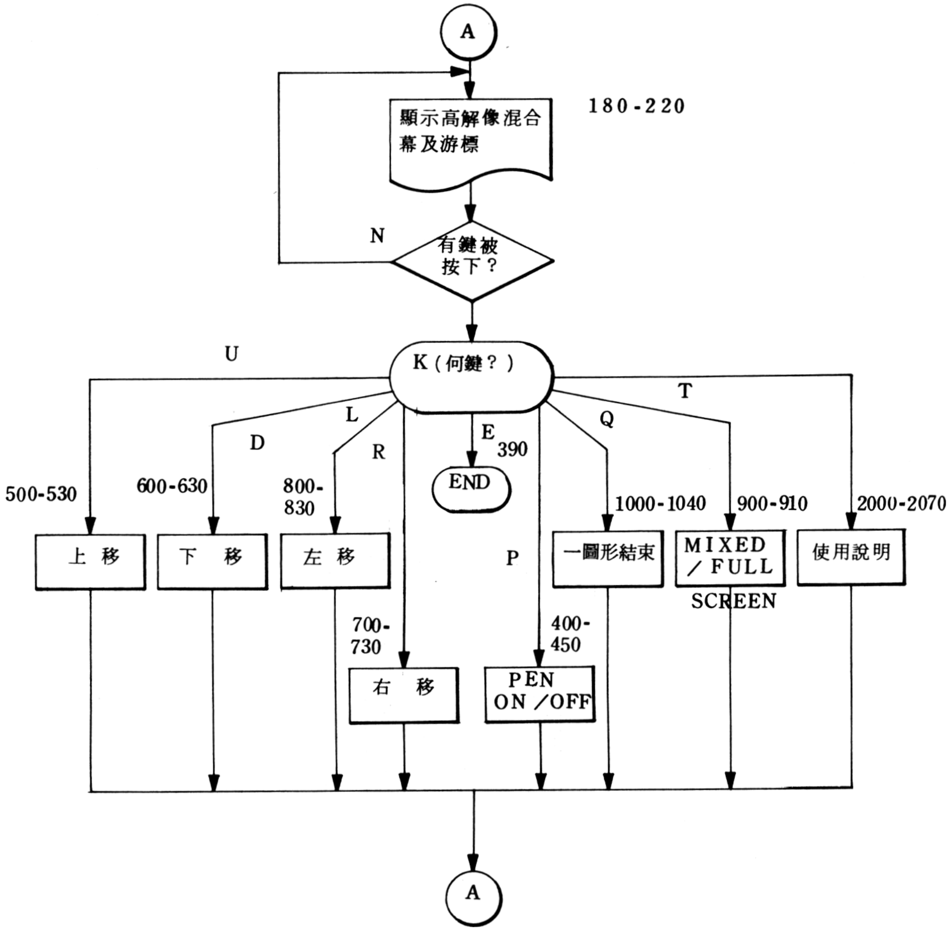
⑽ COUNT ：因圖形表中，每一個圖形代碼由3個位元組成，故記憶體中每一位址，可儲存2至3個圖形代碼。故程式中需要一個計數器COUNT用以指示對一個新的儲存位址而言，已產生幾個圖形代碼。若已有2個或3個則需判斷是否應將這些代碼存入此記憶體位址中。

⑾（OX，OY）：前面已提過變數（X，Y）爲螢幕上游標的座標位置，而此（OX，OY）變數則爲游標的前一個螢幕座標位置。因高解像圖形表中每一個圖形代碼均表示“畫點或不畫點後移動”，即表示先決定畫點或不畫點後，才移動至新位址。因此我們用（OX，OY）記錄此畫點或不畫點的位置，而用（X，Y）代表移動後的螢幕位址。

⑿ T ：變數T用來儲存每一個繪圖動作的圖形代碼。

程式流程圖及說明


開　始
設定各變數初值
顯示使用說明
輸入圖形表儲存位址及包含圖形個數
求出圖形表中第一個圖形之圖形代碼開始儲存位址ADD
A


A
顯示高解像混合幕及游標
180-220
N
有鍵被按下？
K（何鍵？）
U
L
D
R
E
390
END
Q
T
500-530
600-630
800-830
1000-1040
900-910
2000-2070
上 移
下 移
左 移
P
一圖形結束
MIXED / FULL SCREEN
使用說明
700-730
400-450
右 移
PEN ON / OFF
A

(1) 400 － 430 行：將畫點改成不畫點，或將不畫點改成畫點。由變數 PS 記錄其狀態。並在螢幕上顯示目前爲畫點或不畫點之狀態。

(2) 500 － 530 行：將游標上移並根據 PS 之值決定是否須在原位置畫點。T ＝ 0 及 T ＝ 4 分別爲“不畫點上移”及“畫點上移”所應對之圖形代碼（見第四章說明）。

(3) 600 － 630 行，700 － 730 行及800 －830 行：此三部份程式作用與(2)相似，讀者可類推。

(4) 900 － 910 行：轉變混合幕或全圖形螢幕，POKE－ 16302，0 使螢幕成爲全螢幕，POKE － 16301，0 使螢幕成爲圖形文字混合幕。

(5) 1000 － 1040 行：處理一個圖形已製作完畢後之細節。1000 行將最後尚未存入記憶體之圖形代碼存入記憶區中。1020 行判斷是否整個圖形表已製作完畢，是則結束程式，否則重新設定各變數準備製作第二個圖形。1030 行中呼叫副程式 1500 行，此副程式能將下一個要製作的圖形之DIS存入其TA中（見前面變數說明）。

(6) 2000 － 2070 行：顯示使用說明。

(7) 8000 － 8034 行：因一圖形表記憶位址可存 2 到 3 個圖形代碼，此副程式即用以判斷是否應將目前新產生的 2 個或 3 個圖形代碼存入記憶體中。當然不管目前是否要存入記憶體中，均需將圖形代碼按圖形表之形成規則轉換（見第四章說明）。

(8) 9000 行之副程式，眞正將轉換後之圖形代碼（ VL ）存入指定之記憶位址中（ADD）。

```
]LIST

0  HGR
10 ST = 0:PS = 0:COUNT = 1:X = 14
     0:Y = 96
30  GOSUB 2000
170  HOME : INPUT "HOW MANY SHAPE
     S DO YOU WANT TO MAKE?";NUM:
     CNUM = NUM
172  PRINT "WHAT ADDRESS DO YOU W
     ANT TO PUT THESE": INPUT "SH
     APES(RECOMMEND:24576)";BADD
173 CNUM = CNUM - 1:DIS = (NUM +
     1) * 2: GOSUB 1500
174 ADD = BADD + (NUM + 1) * 2: POKE
     BADD,NUM: POKE BADD + 1,0
180  POKE  - 16297,0: POKE  - 163
     04,0
195  VTAB (22): PRINT "X COORD= *
     ";X: PRINT "Y COORD=  *";Y
200  POKE  - 16368,0
205 K =  PEEK ( - 16384): IF K >
     127 THEN 300
210  IF C = 0 THEN C = 1: HCOLOR=
     3: HPLOT X,Y
220 C = 0: HCOLOR= 0: HPLOT X,Y: GOTO
     205
300 K = K - 128: HCOLOR= 3
310  IF K = 85 THEN 500: REM  "U"

320  IF K = 68 THEN 600: REM  "D"

330  IF K = 82 THEN 700: REM  "R"

340  IF K = 76 THEN 800: REM  "L"

350  IF K = 84 THEN 900: REM  "T"

360  IF K = 80 THEN 400: REM  "P"

370  IF K = 32 THEN  GOSUB 2000: GOTO
     180: REM  " "
380  IF K = 81 THEN 1000: REM  "Q
     "
```

```
390  IF K = 69 THEN  CALL 62450: TEXT
     : HOME : END : REM  "E"
395  GOTO 200
400  POKE  - 16301,0
410  IF PS = 0 THEN PS = 1:PS$ =
     "ON": GOTO 430
420 PS = 0:PS$ = "OFF"
430  INVERSE : VTAB 23: HTAB 18: PRINT
     "PLOTTER IS ";PS$: FOR Z = 1
      TO 100: NEXT Z: VTAB 23: HTAB
     18: CALL  - 868: NORMAL : HOME
     : GOTO 195
500 OX = X:OY = Y:Y = Y - 1
510  IF PS = 1 THEN T = 4: HPLOT
     OX,OY: GOTO 530
520 T = 0
530  GOSUB 8000:COUNT = COUNT + 1
     : GOTO 195
600 OX = X:OY = Y:Y = Y + 1
610  IF PS = 1 THEN T = 6: HPLOT
     OX,OY: GOTO 630
620 T = 2
630  GOSUB 8000:COUNT = COUNT + 1
     : GOTO 195
700 OX = X:OY = Y:X = X + 1
710  IF PS = 1 THEN T = 5: HPLOT
     OX,OY: GOTO 730
720 T = 1
730  GOSUB 8000:COUNT = COUNT + 1
     : GOTO 195
800 OX = X:OY = Y:X = X - 1
810  IF PS = 1 THEN T = 7: HPLOT
     OX,OY: GOTO 830
820 T = 3
830  GOSUB 8000:COUNT = COUNT + 1
     : GOTO 195
900  IF ST = 0 THEN ST = 1: POKE
      - 16302,0: GOTO 195
910 ST = 0: POKE  - 16301,0: GOTO
     195
1000  IF COUNT <  > 1 THEN  GOSUB
     9000
1010  GOSUB 9000
1020  IF CNUM = 0 THEN  END
1030 CNUM = CNUM - 1:DIS = ADD -
     BADD: HOME : PRINT "NEXT": FOR
```

```
     Z = 1 TO 100: NEXT Z: HOME :
      GOSUB 1500: CALL 62450
1040 X = 140:Y = 96:COUNT = 0:PS =
     0:ST = 0:C = 0:VL = 0: GOTO
     180
1500 TA = (NUM - CNUM) * 2 + BADD

1510  IF DIS > 255 THEN  POKE TA,
     DIS -  INT (DIS / 256) * 256
     : POKE TA + 1, INT (DIS / 25
     6)
1520  POKE TA,DIS: POKE TA + 1,0
1530  RETURN
2000  TEXT : HOME : HTAB 15: PRINT
     "MAIN MENU": PRINT : PRINT
2010  PRINT "SPACE-BAR -- DISPLAY
      MENU"
2015  PRINT "    'U'     -- MOVE UP
     "
2020  PRINT "    'D'     --MOVE DOW
     N"
2025  PRINT "    'R'     -- MOVE RI
     GHT"
2030  PRINT "    'L'     -- MOVE LE
     FT"
2040  PRINT "    'T'     -- TOGGLE
     SCREEN"
2045  PRINT "                MIXED/F
     ULL"
2048  PRINT "    'P'     -- PEN ON/
     OFF"
2050  PRINT "    'Q'     -- QUIT SH
     APE"
2055  PRINT "    'E'     -- END PRO
     G."
2060  PRINT : PRINT : PRINT : INVERSE
     : PRINT "PRESS ANY KEY TO BE
     GIN!": GET A$: NORMAL
2070  RETURN
8000  VTAB (22): HTAB (20): PRINT
     COUNT;",";T
8005  IF COUNT = 1 THEN VL = T: RETURN

8010  IF COUNT = 2 THEN VL = T *
     8 + VL: RETURN
```

```
8020  IF COUNT = 3 AND PS = 1 THEN
       GOSUB 9000:VL = T:COUNT = C
      OUNT + 1: RETURN
8030  IF COUNT = 3 AND PS = 0 THEN
      VL = T * 64 + VL: GOSUB 9000

8032  IF T = 0 THEN COUNT = 1:VL =
      0
8034  RETURN
9000  VTAB (23): HTAB (20): PRINT
      "VL =";VL: POKE ADD,VL:ADD =
      ADD + 1:COUNT = 0:VL = 0: RETURN


]
```

第三部份
遊戲程式

程式一　APPLE音樂（APPLE MUSIC）

程式二　雷射槍（LAZER BLASTER）

程式三　APPLE COLOR DEMONSTRATION

程式四　太空迷陣（SPACE MAZE）

程式五　繪圖板程式（APPLE PAINTBOX）

程式六　海空大戰（AIRSEA BATTLE）

程式七　打磚塊（LITTLE BRICK OUT）

程式八　魔術方塊（MAD，MAD CUBE）

程式九　金字塔（PYRAMID）

程式十　五子棋（GOBANG）

程式一　APPLE音樂 (APPLE MUSIC)

```
THE HIGH OCTAVE USES THESE KEYS:

  2   3       5   6   7

Q   W   E   R   T   Y   U   I

THE LOW OCTAVE USES THESE KEYS:

  S   D       G   H   J

Z   X   C   V   B   N   M   ,

USE PADDLE 1 FOR ADJUSTING TEMPO
TO BEGIN, PRESS ANY KEY--
```

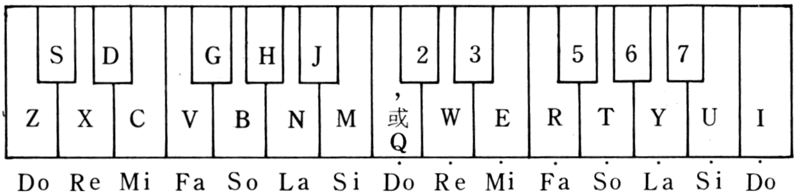
本篇所要介紹的程式是用APPLESOFT BASIC所寫的APPLE MUSIC，利用此程式，將APPLE變成一部簡單的琴鍵式樂器。使遊戲者能在APPLE上彈出樂曲。

只要讀者閱讀過第二部份的"APPLE音效的產生"，相信對我們將要討論的這篇程式便不會感到困難。基本上本程式便是利用到該文所談到的迷你組合語言發聲副程式，及音調數值對應表中部份數據。在本文中對這些重複的觀念不再加以說明，讀者可參考第二部份的"APPLE音效的產生"一章。

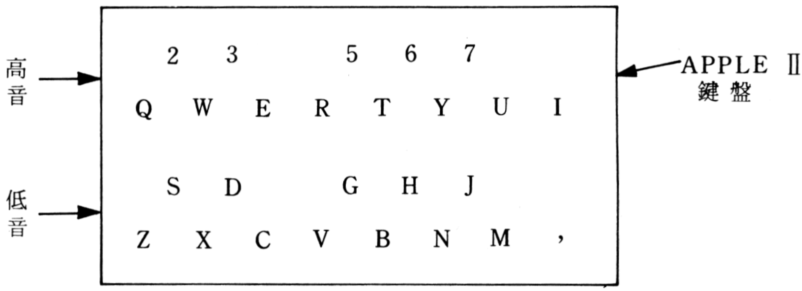
原版的程式是以 Integer Basic 寫成，但顧慮到大多數的讀者可能不太熟悉，故作者將其改成APPLESOFT版本，因此速度上稍微打了些折扣。有興趣的讀者可參考附錄一再改回原版的 APPLE MUSIC。

操作介紹

本程式主要是教讀者如何利用軟體程式將APPLE變成一部樂器。利用鍵盤的按鍵當做琴鍵，使喇叭能發出低八度的音階和高八度音階共25個音（包括半音）。其中對琴鍵的模擬如下圖:



在APPLE的鍵盤上則按高低音階及由左至右代表低音至高音如下圖所示：



其中 ' , ' 和 ' Q ' 鍵發出相同的音Do。

除了由鍵盤選擇音調之外，程式中並用PADDLE 1調整各音的拍子。且為了增加程式的趣味性，程式中並會根據使用者所按下的鍵所代表的高低音符，在螢幕上對應的高低位置顯示該音符的代號；如使用者按下Z鍵，螢幕的第21行會顯示"C"；按下3鍵則螢幕上的14行顯示"D#"（詳細情形後面說明）。

程式中主要變數

(1)A（1－26）：用以儲存發聲機器語言程式所使用的頻率表。但實際上須以32767／A（K）之整數值，才是眞正發聲機器語言程式所使用的頻率值。A（1）至A（26）分別代表低音Do、Re、……Do、Re…Do之值，但A（25）例外，所代表之值與A（13）相同均代表Do之頻率值。A（1）至A（26）與A$中26個字元互相對應。如A$中第三個字元X為Re音，其頻率值即可利用32767／A（3）求出。

(2)A$：為由26個字元所組成的字串變數，各字元與A（1）至A（26）相對應，且按順序分別代表由低至高音所使用的按鍵；但第25個字元"，"例外，並非較'U'所代表的音高半音，而是與'Q'所代表的音相同，是為了配合B$而將其置於'I'之前。

(3)B$：為字串變數，主要供輸出之用。因程式在某一個按鍵被按下後會在螢幕上顯示此鍵所代表的音符，而這些音符便是存在B$中，分別與A$中的每一個字元對應。但由於某些音符須用2個字元表示如D#，C#（半音），有些又只須一個字元表示（全音如C、D），為了方便一律以二個字元代表一個音符，全音只須一個字元者在其後加上一個空白湊成二個字元，因此字串B$中每二個字元對應於A$的一個字元。

(4) C：此變數用以控制螢幕輸出之用，初值爲 0，喇叭每發出一個音，變數 C 加 3，當發出 12 個聲音，螢幕要顯示第 12 個音符前，便將螢幕清除才顯示。即由此變數控制每 11 個音符清洗一次螢幕。

此程式的變數很簡單，嚴格說起來 A(1-26)，A$，B$ 均不是變數（其值不變），而是一些常數。由於變數容易，使整個程式只要基本概念了解了，便不難解讀了。

程式中主要副程式

(1) 1000-2000 行：此副程式用以說明鍵盤上各鍵的使用方式（前面已說明過）

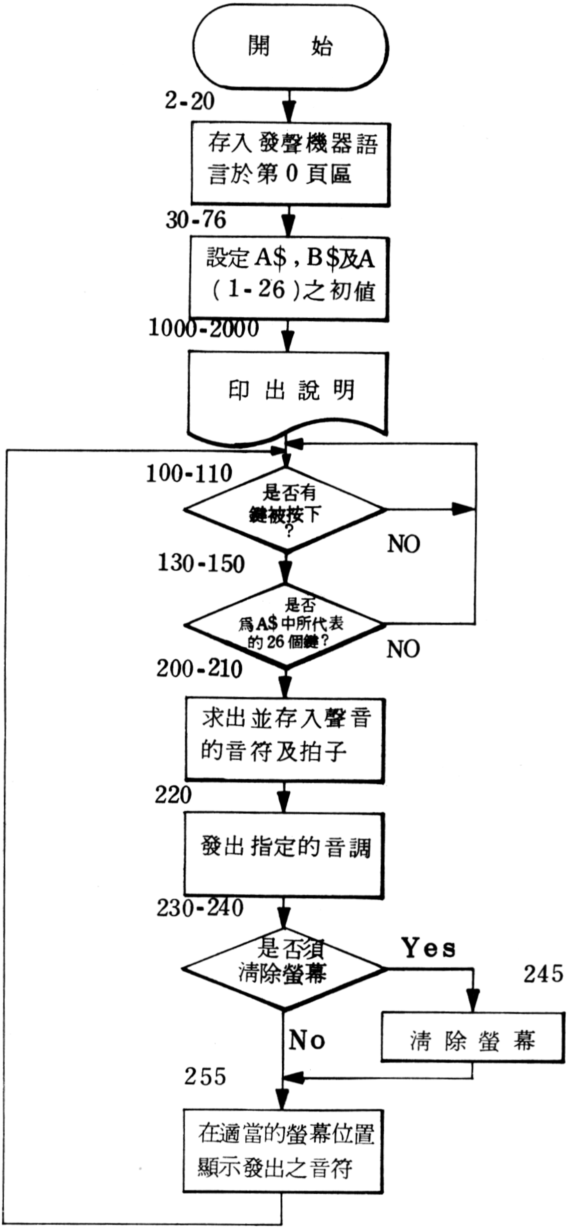
程式的主要流程

（程式的流程圖在下一頁）以下解釋程式中幾個地方：

(1) 1-26 行爲將發聲機語語言存入第 3 頁區。

(2) 100 行與 1980 行的 P=PEEK(−16384)：讀入鍵盤的輸入值，若 P > 128 代表已有鍵被按下（見第一章）。

(3) 120 行及 1990 行的 POKE-16386，0。清除上一次輸入的鍵，使能接受下一次鍵盤輸入。


開　始
2-20
存入發聲機器語言於第0頁區
30-76
設定A$，B$及A（1-26）之初值
1000-2000
印出說明
100-110
是否有鍵被按下？
NO
130-150
是否爲A$中所代表的26個鍵？
NO
200-210
求出並存入聲音的音符及拍子
220
發出指定的音調
230-240
是否須清除螢幕
Yes
245
清除螢幕
No
255
在適當的螢幕位置顯示發出之音符

結語

相信玩過此程式的讀者可能會發現到，彈奏時最不方便的便是拍子不容易控制，事實上利用PADDLE 並不是一個很好的方法。作者建議讀者不妨修改程式使聲音能持續至下一個鍵被按下爲止，如此便可以很容易的控制樂曲的拍子了。

```
]LIST

1  REM  *** APPLE MUSIC ***
2  POKE 776,255
3  POKE 777,255
4  POKE 778,173
5  POKE 779,48
6  POKE 780,192
7  POKE 781,136
8  POKE 774,198
9  POKE 782,208
10  POKE 783,5
11  POKE 784,206
12  POKE 785,9
13  POKE 786,3
14  POKE 787,240
15  POKE 788,9
16  POKE 789,202
17  POKE 790,208
18  POKE 791,245
19  POKE 792,174
20  POKE 793,8
21  POKE 794,3
22  POKE 795,76
23  POKE 796,10
24  POKE 797,3
25  POKE 798,96
30  DIM A(26)
35 B$ = "C C#D D#E F F#G G#A A#B
     C C#D D#E F F#G G#A A#B C C
     "
40 A$ = "ZSXDCVGBHNJMQ2W3ER5T6Y7U
     ,I"
```

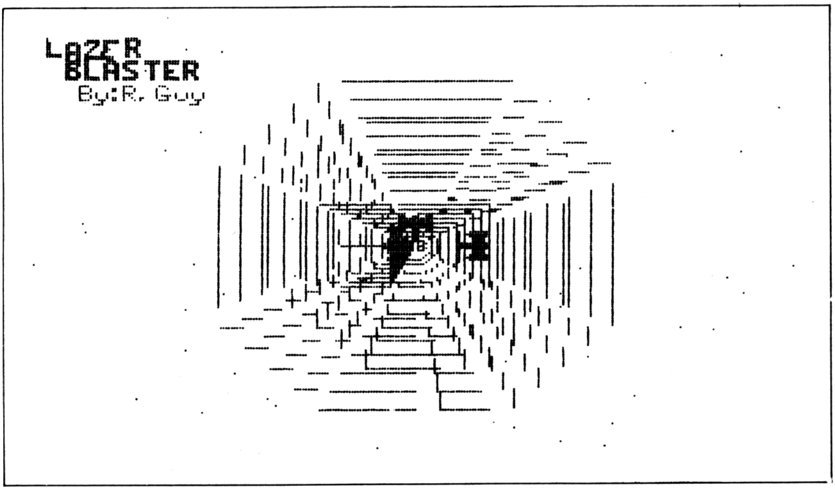
```
50  REM    FREQ TABLE
51 A(1) = 262
52 A(2) = 277
53 A(3) = 294
54 A(4) = 311
55 A(5) = 330
56 A(6) = 349
57 A(7) = 370
58 A(8) = 392
59 A(9) = 415
60 A(10) = 440
61 A(11) = 466
62 A(12) = 494
63 A(13) = 523
64 A(14) = 554
65 A(15) = 587
66 A(16) = 622
67 A(17) = 659
68 A(18) = 698
69 A(19) = 740
70 A(20) = 784
71 A(21) = 831
72 A(22) = 880
73 A(23) = 932
74 A(24) = 988
75 A(25) = 523
76 A(26) = 1047
80  GOSUB 1000
90  CALL  - 936
100 P =  PEEK ( - 16384)
110  IF P < 128 THEN 100
120  POKE  - 16368,0:P = P - 128
130  FOR K = 1 TO 26
140  IF P =  ASC ( MID$ (A$,K,1))
      THEN 200
150  NEXT K
160  GOTO 100
200  POKE 776,32767 / A(K)
210  POKE 777, PDL (1) / 2
220  CALL 778
230 C = C + 3
240  IF C < 37 THEN 250
245 C = 0: CALL  - 936
250  VTAB (26 -  INT (A(K) / 50))
```

```
255  PRINT  MID$ (B$,2 * K - 1,2)

260  GOTO 100
1000  REM  EXPLAIN
1005  CALL  - 936
1010  PRINT "THE HIGH OCTAVE USES
      THESE KEYS:"
1020  PRINT : PRINT
1030  PRINT "  2   3       5   6
       7"
1035  PRINT
1040  PRINT "Q   W   E   R   T
     Y   U   I"
1050  PRINT : PRINT : PRINT
1060  PRINT "THE LOW OCTAVE USES
     THESE KEYS:"
1070  PRINT : PRINT
1080  PRINT "  S   D       G   H
       J"
1085  PRINT
1090  PRINT "Z   X   C   V   B
     N   M   ,"
1100  PRINT : PRINT : PRINT
1105  PRINT "USE PADDLE 1 FOR ADJ
     USTING TEMPO"
1110  PRINT "TO BEGIN, PRESS ANY
     KEY--"
1980  IF  PEEK ( - 16384) < 128 THEN
     1980
1990  POKE  - 16368,0
2000  RETURN
9999  END
10000  REM  DISTRIBUTED BY RAINBO
     W COMPUTING INC. 10732 WHITE
      OAK AVE. TRANADA HILLS CA 9
     1344 (213)-360-2171
10005 MOD IF IEDBYTZ ON GSHIELIN
```

程式二　雷射槍

(LAZER BLASTER)



APPLESOFT BASIC 語言的速度雖不及機器語言的速度快，但往往利用一些技巧也能達到很好的效果，雷射槍程式便是很好的例子。

此程式利用APPLE的高解像畫面第二頁畫出圖形，而將圖形表存在高解像第一頁區。遊戲的主要進行方式爲：首先在螢幕中央出現一座雷射槍座，位於槍座上的雷射槍可利用 PADDLE（0）控制此雷射槍向上、下、左、右四個方向移動，對位於上、下、左、右四個可能出現敵方目標物的方向進行射擊，但並非每

個方向均會有敵目標，可能僅出現一個目標，雷射槍瞄準好後，按下PUSHBUTTON＃0 便可發射雷射光將敵人目標摧毀。但在雷射槍向目標瞄準的同時各敵方目標會由其所在方向，向雷射槍座逼近，若雷射槍無法在每一個敵人目標到達雷射槍座之前，將所有敵人目標摧毀，本身就會遭敵人摧毀。若能在敵方目標撞到雷射槍座前將其盡數摧毀，便能按所摧毀目標的種類而得到不同的分數，但在下次目標再出現之時，其移動速度會較原來的速度快（直到某一限度）。

得分之計算

此遊戲分為1-3 三種難易程度，程式中根據使用者所選擇的難易程度，決定敵方目標物的移動速度和得分多寡。但每當遊戲者將螢幕上敵方目標全部摧毀後，程式會自動加快下一次出現的敵目標的移動速度，使遊戲愈來愈難。但至於得分的計算，則完全根據遊戲者當初所選擇的難易程度與敵目標的種類而定，而與敵目標移動速度的快慢無關。其分數對照表如下：

敵方目標	難度1	難度2	難度3
大型火箭	5	10	15
小型火箭	10	20	30
小遊星	15	30	45

且程式會記錄目前所得的最高分。

關於圖形表

在遊戲正式開始前，須將圖形表存入＄2000至＄222D 的位址中。圖形表包括11個圖形：

1. 雷射槍座



2. 大型火箭



3. 小型火箭



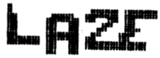
4. 小遊星



5. 雷射槍



6. 文字圖形



7. 文字圖形



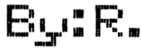
8. 文字圖形



9. 文字圖形



10. 文字圖形（細體字）



11. 文字圖形（細體字）



其中圖形 6-11 只是用來在遊戲畫面的左上角顯示出遊戲名稱與撰寫者姓名。程式中所不斷使用的圖形則爲 2. 3. 4. 5. ，讀者可在程式中發現。

在此程式假設圖形表已存在磁碟上，檔案名稱爲LB.O，因此在程式開頭 60 行，檢查 $E 9 是否爲$ 20 便可知道圖形表是否已存入$ 2000－$ 222 D位址中，如此只有在頭一次玩此遊戲時由磁碟中讀入圖形表，免得每一回合遊戲均須讀入一次。對於沒有磁碟機的讀者，便須自己動手由磁帶上讀入$ 2000－$ 222 D之位址，並應將程式第 60 行改爲：

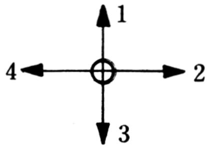
```
60 TEXT : HOME
```

程式中主要變數

(1)L ：主要用以決定敵方目標每次向雷射槍座的逼近距離。L之初值由遊戲者所選擇的難易程度而給予1-3之值，每次新的敵人目標出現後L會自動加上0.15使敵目標逼近速度愈來愈快，但L之極大值爲5。

(2) I L：記錄使用者所選擇的難易程度（1至3），供計算得分之用。

(3) LGUN：其值有1.2.3.4.四種，用以表示雷射槍所面對的方向如圖右：

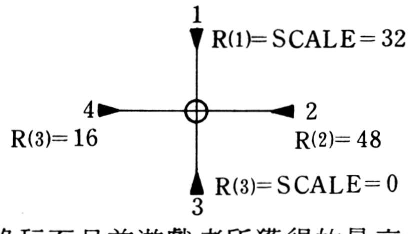


，其值由POL（0）之值所決定。

(4) SH（1-4）：爲一陣列，SH（n）表示方向n（上圖）的敵目標之圖形爲圖形表的第幾個圖形，其值爲0、2、3或4。

(5)RO（1-4）：因敵目標必須面對雷射槍座逼近，故必須對各圖形表的圖形加以不同角度的旋轉（利用SCALE =），而RO（n）即表示在方向n時，所需要的旋轉值。因在繪製圖形表時圖形均面對上方故各方向的旋轉值給定如下圖：

（見第4章說明）



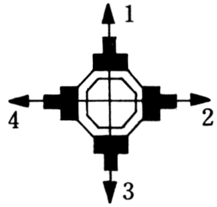
(6)HS：Hight Score ，記錄玩至目前遊戲者所獲得的最高分數。

(7) SC：Score ，本回合所得到的分數。

(8) SH：用以判斷螢幕上是否仍有敵目標存在，若目標均遭擊毀，則SH之值便爲0，（SH=SH（1）+SH（2）+SH（3）+SH（4））。

程式主流程所使用的副程式

(1) 195，200，205，210 行：根據LGUN 之值，決定雷射槍之方向後，呼叫 195，200，205，或 210 行以繪出雷射槍之位置圖形如圖右：（因雷射槍原圖形面對方向 2，故方向 2 之ROT ＝0，與敵方目標之旋轉值不同）。



方向	座標	ROT=
1	(140,87)	48
2	(149,95)	0
3	(140,105)	16
4	(131,96)	32

(2) 230-265 行：根據 LGUN 之值，可決定出雷射槍之位置，再呼叫 230，240，250 或 260 行之副程式，用以判別雷射槍發射的方向上是否有敵目標存在（IF SH（LGUN）< 2…），若無則僅畫出雷射光及發出聲音；若有須呼叫 270行之副程式（見下）。

(3) 270-275 行：此副程式，從雷射槍至目標物間畫出一道雷射光，消除敵目標，再以SCALE＝2 重新畫出該敵目標，並發出聲音，再將雷射光及該敵目標清除，以模擬雷射槍發射擊中目標，目標爆炸，至消失的過程。最後計算得分，並標示該方向上已無敵人目標（SH（LGUN）＝0）。

(4) 280-315行：內含 280，290，300 及 310 行的四個副程式，分別用來移動上、右、下、左方向上之敵目標，即求出新的位置後在該位置重新繪出該目標，而新位置（或移動距離）由L 決定。且因橫座標較縱座標長，爲使目標能幾乎同時到達雷射槍座，水平方向目標每次逼近距離爲 7 ＊ L而垂直方向僅爲5＊L 。最後此副程式比較此新位置是否已到達雷

射槍座，未到則RETURN 繼續，若已到達則至 340 行副程式進行處理（但在GOTO 340之前有一POP動作，因目前執行的副程式是被別的程式所呼叫，在執行目前的副程式前，已將RETURN ADDRESS存入（PUSH)堆列(Stack)中。而在此副程式GOTO 340 之後，一切重新開始，不再利用此RETURN ADDRESS以回到原來呼叫的程式，故須 POP，將在堆列中的RETURN ADDRESS 除去，否則會發生錯誤）。

(5) 320-325 行：發出雷射光射擊之聲音，此副程式呼叫位於位址 778 的發聲機器語言程式以發聲（此機器語言發聲程式，前面已介紹過不再贅述）。

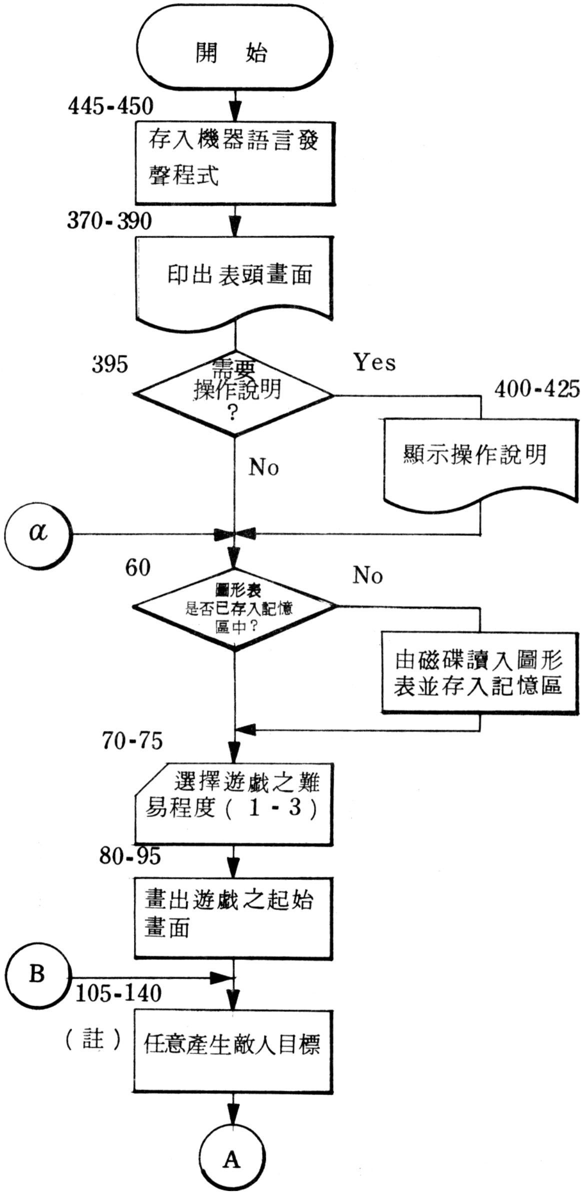
(6) 330-335 行：發出雷射槍座爆炸的聲音。

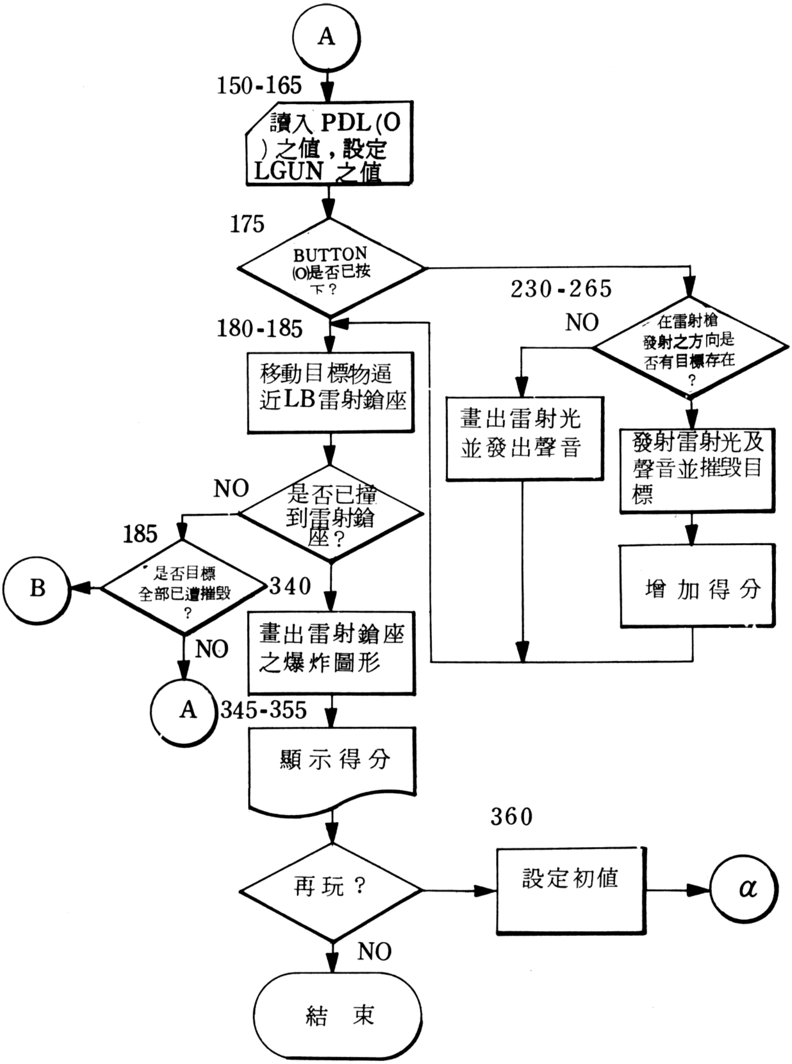
(7) 435 行：此副程式可將A$字串之內容印在螢幕的中央位置。此種方式，可以很容易的印出整齊的標題畫面。

(8) 445-450 行：將機器語言副程式存入第三頁記憶區中。

程式流程

對於程式流程我們以流程圖方式說明，須進一步解釋者，再用註解說明：


開　始
445-450
存入機器語言發聲程式
370-390
印出表頭畫面
395
需要操作說明？
Yes
No
400-425
顯示操作說明
α
60
圖形表是否已存入記憶區中？
No
由磁碟讀入圖形表並存入記憶區
70-75
選擇遊戲之難易程度（1-3）
80-95
畫出遊戲之起始畫面
B
105-140
（註）
任意產生敵人目標
A



（註）對於敵方目標物的產生，程式中利用 RND ，在任意方向，上產生任意種類的目標物。

```
]PR#0
]LIST

10  REM  ************************
     *
20  REM  **   LASER - BLASTER   *
     *
25  REM  **   BY   RUDY A.GUY   *
     *
30  REM  **     JUNE 7, 1981    *
     *
35  REM  **    COPYRIGHT 1981   *
     *
40  REM  ************************
     *
50  GOSUB 445
55  GOSUB 370
60  TEXT : HOME : IF  PEEK (233) <
      > 32 THEN  PRINT  CHR$ (4)"
     BLOAD LB.O"
65  POKE 232,0: POKE 233,32: VTAB
     12
70  INPUT "SELECT SKILL LEVER (1-
     3) ";L: IF L < 1 OR L > 3 THEN
     60
75 IL = L
80  REM  << INITIALIZE HI-RES SCR
     EEN>>
85  HGR2 : ROT= 0: SCALE= 1: HCOLOR=
     3: DRAW 1 AT 140,96
90  DRAW 6 AT 5,5: DRAW 7 AT 33,5
     : DRAW 8 AT 12,15: DRAW 9 AT
     40,15: DRAW 10 AT 17,32: DRAW
     11 AT 45,32
95  FOR I = 1 TO 100: HCOLOR=  RND
     (1) * 7 + 1: HPLOT  RND (1) *
     280, RND (1) * 192: NEXT : REM
      STARTS
100  REM  ALIEN SHIPS
105  FOR I = 1 TO 4: READ XI(I),Y
     I(I),RO(I):X =  INT ( RND (1
     ) * 10 + 1)
110  IF X / 2 <  >  INT (X / 2) THEN
     130
115  IF X / 2 =  INT (X / 2) THEN
     SH(I) = 2
```

```
120 X =  INT ( RND (1) * 10 + 1):
     IF X = 7 THEN SH(I) = 3
125  IF X = 10 THEN SH(I) = 4
130  NEXT : HCOLOR= 3
135  FOR I = 1 TO 4: IF SH(I) > 0
      THEN  ROT= RO(I): DRAW SH(I
     ) AT XI(I),YI(I)
140  NEXT
145  REM  << READ PADDLE & ASSIGN
      LAZER POSITION >>
150 P =  PDL (0): IF L > 5 THEN L
      = 5
155 LGUN =  INT ((P + 45) / 75): IF
     LGUN = 0 THEN LGUN = 1
160  IF Y > 0 THEN  HCOLOR= 0: ROT=
     R: DRAW 5 AT X,Y: HCOLOR= 3
165  ON LGUN GOSUB 195,200,205,21
     0
170  REM  << CHECK FOR BUTTON (0)
      >>
175  IF  PEEK ( - 16287) > 127 THEN
      GOSUB 215
180  FOR I = 1 TO 4: IF SH(I) > 1
      THEN  ROT= RO(I): HCOLOR= 0
     : DRAW SH(I) AT XI(I),YI(I):
      HCOLOR= 3: ON I GOSUB 280,2
     90,300,310
185  NEXT : FOR I = 1 TO 4:SH = S
     H + SH(I): NEXT : IF SH = 0 THEN
      RESTORE :L = L + .15: HCOLOR=
     0: ROT= R: DRAW 5 AT X,Y: HCOLOR=
     3: GOTO 105
190 SH = 0: GOTO 150
195  ROT= 48: DRAW 5 AT 140,87:X =
     140:Y = 87:R = 48: RETURN
200  ROT= 0: DRAW 5 AT 149,95:X =
     149:Y = 95:R = 0: RETURN
205  ROT= 16: DRAW 5 AT 140,105:X
      = 140:Y = 105:R = 16: RETURN
210  ROT= 32: DRAW 5 AT 131,96:X =
     131:Y = 96:R = 32: RETURN
215  REM  <<FIRE LAZER >>
220  ON LGUN GOSUB 230,240,250,26
     0
225  RETURN
```

```
230  IF SH(LGUN) < 2 THEN  HPLOT
     140,82 TO XI(LGUN),YI(LGUN):
      GOSUB 325: HCOLOR= 0: HPLOT
     140,82 TO XI(LGUN),YI(LGUN):
      HCOLOR= 3: RETURN
235 HX = 140:HY = 82: GOSUB 270: RETURN
240  IF SH(LGUN) < 2 THEN  HPLOT
     153,95 TO XIX(LGUN),YI(LGUN)
     : GOSUB 325: HCOLOR= 0: HPLOT
     153,95 TO XI(LGUN),YI(LGUN):
      HCOLOR= 3: RETURN
245 HX = 153:HY = 95: GOSUB 270: RETURN
250  IF SH(LGUN) < 2 THEN  HPLOT
     140,109 TO XI(LGUN),YI(LGUN)
     : GOSUB 325: HCOLOR= 0: HPLOT
     140,109 TO XI(LGUN),YI(LGUN)
     : HCOLOR= 3: RETURN
255 HX = 140:HY = 109: GOSUB 270:
      RETURN
260  IF SH(LGUN) < 2 THEN  HPLOT
     127,96 TO XI(LGUN),YI(LGUN):
      GOSUB 325: HCOLOR= 0: HPLOT
     127,96 TO XI(LGUN),YI(LGUN):
      HCOLOR= 3: RETURN
265 HX = 127:HY = 96: GOSUB 270: RETURN
270  HPLOT HX,HY TO XI(LGUN),YI(L
     GUN): ROT= RO(LGUN): HCOLOR=
     0: DRAW SH(LGUN) AT XI(LGUN)
     ,YI(LGUN): SCALE= 2: HCOLOR=
     3: DRAW SH(LGUN) AT XI(LGUN)
     ,YI(LGUN)
275  GOSUB 325: HCOLOR= 0: HPLOT
     HX,HY TO XI(LGUN),YI(LGUN): DRAW
     SH(LGUN) AT XI(LGUN),YI(LGUN
     ):SC = SC + (SH(LGUN) * IL *
     5):SH(LGUN) = 0: SCALE= 1: RETURN
280 YI(I) = YI(I) + (5 * L): DRAW
     SH(I) AT XI(I),YI(I): IF YI(
     I) =  > 87 THEN  POP : GOTO
     340
285  RETURN
290 XI(I) = XI(I) - (L * 7): DRAW
     SH(I) AT XI(I),YI(I): IF XI(
     I) <  = 149 THEN  POP : GOTO
     340
```

```
295  RETURN
300 YI(I) = YI(I) - (5 * L): DRAW
     SH(I) AT XI(I),YI(I): IF YI(
     I) <  = 105 THEN  POP : GOTO
     340
305  RETURN
310 XI(I) = XI(I) + (7 * L): DRAW
     SH(I) AT XI(I),YI(I): IF XI(
     I) >  = 131 THEN  POP : GOTO
     340
315  RETURN
320  REM  LAZER SOUND
325  FOR I = 12 TO 10 STEP  - 1: POKE
     776,7: POKE 777,I: CALL 778:
      NEXT : RETURN
330 SOUND FOR DESTRUCTI ON
335  POKE  - 16336, PEEK ( - 1633
     6): POKE  - 16336, PEEK ( -
     16336): POKE  - 16336, PEEK
     ( - 16336): RETURN
340  FOR I = 1 TO 10: SCALE= I: DRAW
     1 AT 140,96: GOSUB 335: NEXT
345  TEXT : HOME : IF SC > HS THEN
      VTAB 7: PRINT "NEW HIGH SCO
     RE : ";: FLASH : PRINT SC: NORMAL
     :HS = SC: GOTO 355
350  VTAB 7: PRINT "HIGH SCORE :"
     HS: VTAB 9: PRINT "YOUR SCOR
     E : "SC
355  VTAB 20: PRINT "ANOTHER TRY
     ?";: GET A$: IF A$ = "N" THEN
      END
360 SC = 0:R = 0:X = 0:Y = 0:IL =
     0: RESTORE : GOTO 60
365  DATA  140,12,32,260,96,48,14
     0,178,0,16,96,16
370  REM  TITLE
375  TEXT : HOME
380  VTAB 10:A$ = "L A Z E R - B
     L A S T E R": FLASH : GOSUB
     435: NORMAL
385  VTAB 12:A$ = "BY: RUDY A. GU
     Y  JUNE 7, 1981": GOSUB 435
390  VTAB 14:A$ = "(C) COPYRIGHT
     1981 - MICRO-SPARC INC.": GOSUB
     435
```

```
395  VTAB 23: PRINT "INSTRUCTIONS
      (Y/N)?";: GET A$: PRINT : IF
     A$ = "N" THEN  RETURN
400  HOME :A$ = "L A Z E R - B L
     A S T E R": INVERSE : GOSUB
     435: NORMAL : PRINT
405  PRINT "   THE OBJECT OF 'LAZ
     ER-BLASTER' IS TO SHOOT AS M
     ANY ALIENS AND ASTEROIDS AS
       POSSIBLE. GAME PADDLE (0)
     IS USED TO    TURN AND FIRE
     THE LAZER."
410  PRINT : PRINT "   THERE ARE
     3 LEVERS OF PLAY FROM WHICH
     TO CHOOSE.(LEVER 1 IS THE EA
     SIES.)    AS YOUR SCORE INCR
     EASES, SO WILL THE    SPEED
     OF THE ALIEN SHIPS. SCORING
     IS AS FOLLOWS:"
415  PRINT : PRINT "ATTACKER
          LEV.1  LEV.2  LEV.3   -
     ----------------------------
     -----------LARGE SHIP
        10      20      30"
420  PRINT "SMALL SHIP            1
     5      30      45    ASTEROID
                20      40      60"
425  VTAB 23: PRINT "PRESS ANY KE
     Y TO BEGIN.":X =  PEEK ( - 1
     6384): IF X > 127 THEN  POKE
      - 16368,0: RETURN
430  GOTO 425
435  HTAB 20 -  LEN (A$) / 2: PRINT
     A$: RETURN
440  REM  << SOUND ROUTINE >>
445  POKE 776,255: POKE 777,255: POKE
     778,173: POKE 779,48: POKE 7
     80,192: POKE 781,136: POKE 7
     82,208: POKE 783,5: POKE 784
     ,206: POKE 785,9: POKE 786,3
     : POKE 787,240
450  POKE 788,9: POKE 789,202: POKE
     790,208: POKE 791,245: POKE
     792,174: POKE 793,8: POKE 79
     4,3: POKE 795,76: POKE 796,1
     0: POKE 797,3: POKE 798,96: RETURN
```

```
*2000.2230

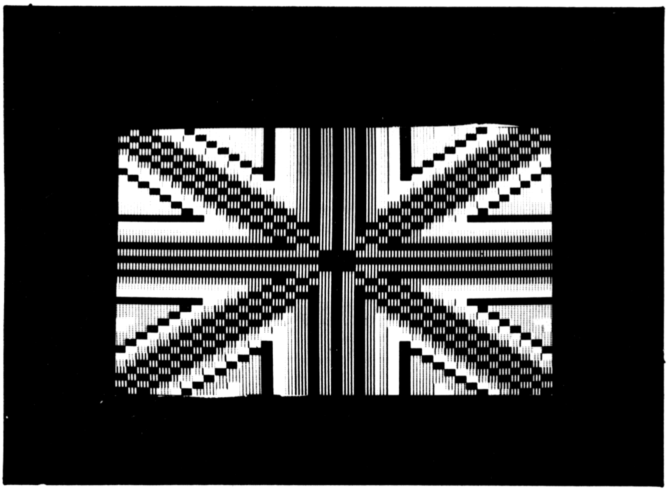
8192
2000- 0B 00 18 00 67 00 A8 00
2008- C4 00 DE 00 E9 00 35 01
2010- 4B 01 A6 01 DE 01 0F 02
2018- 80 80 80 80 80 80 80 28
2020- 2D 15 15 15 15 15 36 36
2028- 36 1E 1E 1E 1E 1E 3F 3F
2030- 3F 07 38 38 38 38 20 24
2038- 24 64 0C 0C 0C 0C 2D 16
2040- 2D 75 0E 0E 0E 36 36 1E
2048- 1E 1E 1E 3F 3F 07 38 38
2050- 38 20 24 64 0C 0C 0C 96
2058- 1A 36 36 2D 0D 2D 05 E0
2060- 3F 4C E1 3F 36 26 00 80
2068- 80 80 80 80 80 A8 36 36
2070- 2E 2D 64 15 36 36 36 3E
2078- 24 24 24 3C 36 36 36 3E
2080- 80 20 24 37 36 37 3F 2C
2088- 25 24 37 3E 37 3F 36 3F
2090- 2C 3C 24 24 64 36 36 2E
2098- 24 24 15 36 2E 24 2C 36
20A0- 25 24 24 24 37 36 26 00
20A8- 80 A8 36 36 AE 15 15 3E
20B0- 3F 3F 3F 67 2D 2D 2D 3C
20B8- 3F 3F 0C 24 24 24 35 36
20C0- 36 2E 04 00 2D 05 60 2D
20C8- 0C E4 E7 3F 3F 37 3F F7
20D0- 17 2E 2D 2D 28 B5 D8 D8
20D8- 0E 35 2D 2D 04 00 24 2C
20E0- 36 2D 36 27 37 36 27 24
20E8- 00 36 36 36 36 2D 2D 25
20F0- 3F 3F 24 24 6C 49 11 36
20F8- 36 36 2E 24 24 24 24 2D
2100- AD 3F 77 31 3E 37 2D 36
2108- 2E 24 24 24 05 40 2E A4
2110- 2C 2E 2C 35 37 F5 37 37
2118- 37 37 2E 2C 2E 2C 2E AC
2120- 21 24 24 24 2C 2D 2D 3E
2128- 3F 37 36 3D 3E 37 2E 2D
2130- 35 3F 3F 04 00 31 36 36
2138- 36 2E 24 24 24 24 35 25
2140- 35 35 37 F5 3F 2E 35 77
2148- 35 27 00 36 36 36 36 25
```

```
2150- 24 24 24 2C 2D 15 3F 77
2158- 29 1E 3F 2E AD 37 35 3F
2160- 37 2D 0D 80 80 80 80 80
2168- 80 40 36 36 36 36 2D 2D
2170- 25 3F 3F 24 24 24 6C 49
2178- 11 36 36 36 2E 24 24 24
2180- 24 2D AD 3F 77 29 3E 9F
2188- 2D 35 3F 77 29 3E 2E 0D
2190- 2D 2D 05 20 E4 3F 3F 20
2198- 64 2D 2D 3E 3F 37 2E 15
21A0- 2E 36 3F 3F 04 00 31 25
21A8- 35 36 36 36 2E 24 24 24
21B0- 24 35 25 0D 36 36 36 36
21B8- 2D 2D 25 3F 3F 24 2D 3C
21C0- 27 24 2C 2E 2C 2E 6C 36
21C8- 36 36 36 25 24 24 24 2C
21D0- 2D 15 3F 77 29 3E 3E 37
21D8- 2D 3E 0E 35 27 00 24 24
21E0- 24 2D AD F6 3F 4E 31 1E
21E8- 3F 4E 09 80 80 40 36 0E
21F0- 35 DF 2A 2D 05 80 28 20
21F8- 24 48 35 B7 2A 3E 4E 09
2200- 20 24 24 2C 2D 15 F6 3F
2208- 0E 15 15 0D 25 27 00 20
2210- 24 64 2D 15 96 39 3F 97
2218- 2D 25 0D 40 36 0E 2D 05
2220- 20 6C 36 0E 35 DF 2A 2D
2228- 05 80 60 24 04 00 A5 44
2230- 4C
```

8752

程式三

APPLE COLOR DEMONSTRATION



此程式正如名稱所示爲一示範（或說明）程式，用以說明 APPLE Ⅱ 在低解像畫面時所擁有的 16 種顏色及各顏色的代碼，並利用二個副程式在螢幕上示範顏色的變化。

此程式並不困難，對於程式設計不是很有經驗的讀者，此程式更能提供一些程式設計的觀念和技巧。程式主要分爲四大部份進行即：

1. STANDARD COLOR NAMES：在低解像螢幕上以垂直長方形顯示出 16 種顏色。並在各顏色下註明此顏

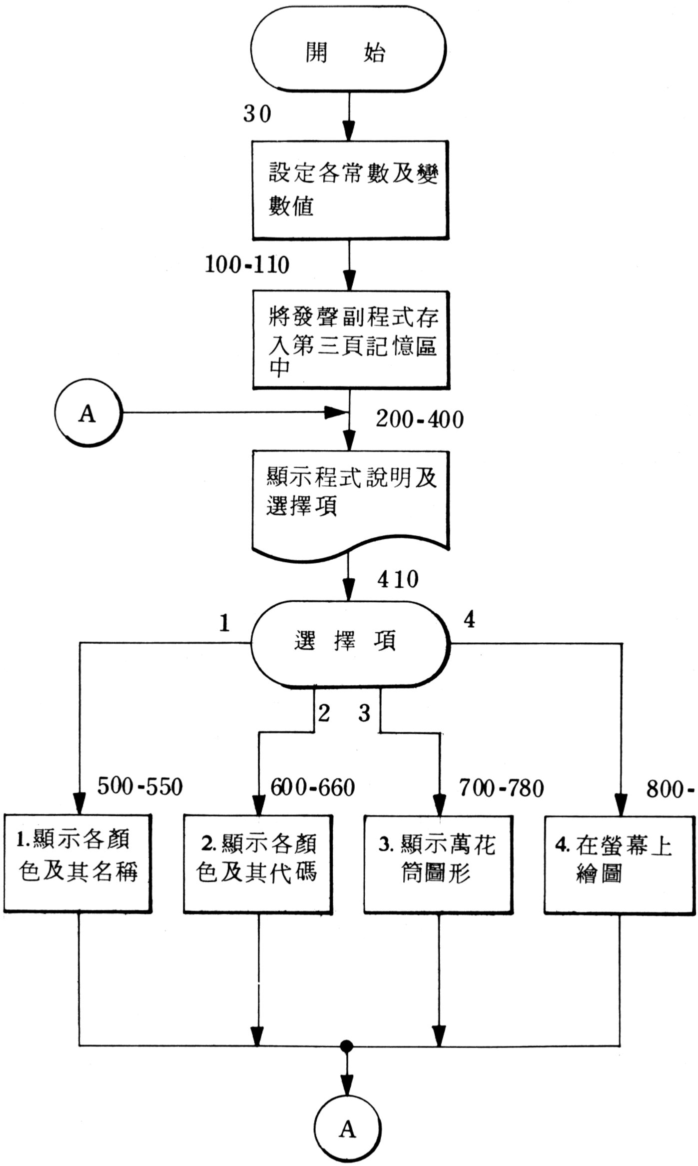
色之英文名稱。

2. STANDARD COLOR NUMBERS：在低解像螢幕上以垂直長方形顯示出 16 種顏色。並在各顏色下註明此顏色之代碼。

3. KALEIDOSCOPE：此部份為整個程式中最精彩的部份，在低解像螢幕上模擬萬花筒的顏色及圖形變化。也是整個程式中最難的一部份。

4. SKETCHING SCREEN：利用 PADDLE及PUSHBUTTON 使遊戲者可在螢幕上畫出顏色不同的圖形，此外程式會配合各圖形的畫點位置產生不同的聲音，以增加趣味性。

由於程式主要分為上述四大部份進行，因此本文說明也分為四大部份說明。但對於共用的副程式，則先討論。下面是整個程式的流程圖：


開 始
30
設定各常數及變數值
100-110
將發聲副程式存入第三頁記憶區中
A
200-400
顯示程式說明及選擇項
410
1
選 擇 項
4
2
3
500-550
600-660
700-780
800-
1.顯示各顏色及其名稱
2.顯示各顏色及其代碼
3.顯示萬花筒圖形
4.在螢幕上繪圖
A

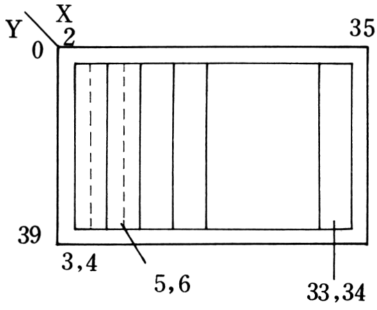
程式之共用副程式

(1) 1500 行：此副程式是使喇叭發出指定的聲音，在程式一開頭 100-110 行便已將發聲程式存入第三頁記憶區。此副程式只是將發聲程式所需的輸入參數（聲音高低及聲音長短，前者須存入 $6 位址，後者存入 $7）P（PITCH）及D（DURATION）存入指定位址，然後呼叫此發聲副程式以發出聲音。

(2) 2000 行：此副程式主要是用在螢幕上顯示字串之用。使A$ 顯示在螢幕上某一列的中央位置。這是一種很簡單又很實用的方法。

(3) 3000-3010 行：此副程式能在低解像混合幕上畫出 16 個不同顏色的長方形，以顯示APPLE II在低解像圖形中所能產生的顏色。3000 行中的COLOR = I / 2 使得每一塊長方形的寬爲二單位。3010 行只是在螢幕上畫出灰色的邊框，可以右圖表示：

此副程式是利用低解像圖形文字混合幕，故保留螢幕上最下面 4 行供顯示文字。



(4) 4000-4020 行：此副程式也是在螢幕上顯示字串之用。能使 A$ 內容的各字元以間隔一個空白的方式顯示在螢幕上。在此所用的方法爲將A$ 內之字元逐字用 MID$ 取出，以間隔一個位置的方式，顯示在螢幕上。由於是逐字印出因此4010 行中的 PRINT ，MID$（A$，I，1）後需用分號而不能

用冒號，如此使得每一個A$ 被印完後需加上PRINT指令使跳行（如 510-530行）。其實較好的方法是在A$ 一印完立即跳行，即在副程式中完成此跳行動作（可在 4020 行 NEXT I之後加上 PRINT：），則主程式便不需考慮加上額外的 PRINT指令，使程式更加清楚更有結構。

程式流程說明

(1) 500-550行：此部份能在低解像螢幕上顯示出 16 種低解像畫面所擁有的顏色及各顏色的名稱。首先呼叫 3000 行的副程式印出圖形，再分別呼叫 4000 行的副程式在各顏色下印出其名稱。由於只有四行文字行，且名稱須以垂直方式顯示即黑色要顯示為“B
L
A
K”而非“BLAK”故 510至 540 行的各 A$ 均經過排列。顯示結束後，可按下任一鍵回到說明部份（200）行。

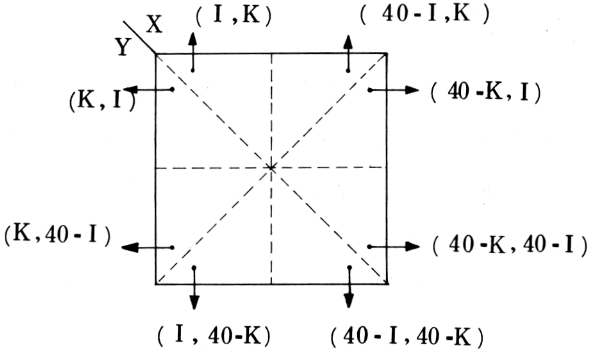
(2) 600-660行：此部份也是在低解像螢幕上顯示出16種低解像畫面所擁有的顏色及各顏色的代碼。首先仍先呼叫 3000 行的副程式印出圖形。再在圖形之下方文字行顯示各顏色的代碼。如圖右：

X=	3,4	5,6	7,8		31,32	33,34
	0	1	2 …… 3 ……		14	15

程式中用到許多HTAB指令以決定字元輸出位置。因此讀者應弄清楚 HTAB 3並不是指在低解像X座標為 3之點，而是在X座標為 2之點。因此上圖中 0，1，…，15 數字出現的位置正如圖所示。如在 620 行若 I＝0，HTAB 為 4 ，但 '0'顯示在第 3 行而非第 4 行。

顯示結束後，可按下任一鍵回到說明部份。

(3) 700-780 行：用以顯示萬花筒圖形變化。利用 760 及 770 行執行眞正畫點動作，對每一個點座標（I，K）在螢幕上畫出對稱的八點，而此點的顏色已事先在 750 求出，亦即一種顏色在螢幕上一次畫出對稱的 8 個點，如下圖所示：



圖中虛線表示各點的對稱軸。

而點座標的產生，與顏色的改變則完全受三個巢狀 FOR … NEXT 迴圈的控制：

```
FOR W = 3 TO 50
      FOR I = 1 TO 19
           FOR J = O TO 19
                      ⋮
                K = I + J
                      ⋮
              NEXT J
           NEXT I
       NEXT W
```

即　I = 1　時　K = 1　至 20

　　I = 2　時　K = 2　至 21

　　⋮　　　⋮　　　⋮

　　I = 19　時　K = 19　至 38

加上一個（I，K）座標便畫出互相對稱的8個點，且每一次I值加1時便畫出20個點，整個圖形便漸漸由外向內一圈一圈的畫出。且由於每一圈圖案固定畫出20個點及畫點顏色的改變，便使得圖形愈向中央時，也愈顯得有向外擴展的趨勢。這種方式很難用言語述敍，讀者可在程式中加入下面一行：

775　FOR SL＝O TO 500：NEXT SL

使繪圖的速度減慢便可很清楚的知道整個圖案形成的基本原理。每一個I值在螢幕上向內畫出一圈圖形，故當I自0變成19時便已畫完整個螢幕一次；而每一個W值便需畫滿螢幕一次（畫點方式完全相同），故知整個萬花筒圖案變化完成共需畫滿螢幕48次，時間相當久。故爲了怕使用者不耐煩，程式中便在700行測試是否有鍵盤上的鍵被按下，有則停止圖案的顯示，跳至200行。當然此測試必需隨時進行，否則使用者可能會感覺到“鍵按了也沒什麼反應”，因此必須每畫8點便測試一次，即置於最內的迴圈中。

程式中對稱畫點方式是最主要的觀念，但顏色的變化亦不可忽視。在750行設定顏色的指令中，包括I，J，W 3個變數，因I，J，W 值不斷改變，便可使顏色呈某種方式改變，再經由對稱畫點的方式，便可造出極優美的圖案。此外讀者應注意750行中一定要包括三個變數，例如若未含W變數，便會使得相同圖形呈現48次而已。

當然顏色的設定函數，讀者可自行變化，均可產生優美的圖形。如讀者可將750行改成

750 COLOR＝J＊3／I＋ I＊W／12

或　750 COLOR＝J＊4／（I＋4）＋I＊W／6 …等

使螢幕上產生不同的圖案。讀者何不試試！

(4) 800-1020 行：程式中G = − 16287 爲PUSHBUTTON # 0讀取位址，H = − 16286 爲PUSHBUTTON # 1 之讀取位置，E = − 16384 ，爲鍵盤讀取位址。程式列出說明後便由900 行測試是否 PUSHBUTTON # 0或# 1 被按下，直到任一 PUSBUTTON被按下，才會正式開始執行。否則由910 行測試是否有鍵被按下以跳回程式之200 行說明。940及950行分別由PADDLE # 0 及 # 1 求出畫點座標，並在960行利用所讀取之 PADDLE 值，求出發聲之音調，至於聲音長短則固定D = 3，呼叫發聲程式發出高低不同的聲音。

在程式進行中，可按下任一鍵停止此部份程式（由990行測試）或按下PUSHBUTTON # 0重新畫圖，或按下PU-SHBUTTON # 1改變畫點的顏色（由1000及1010 行測試）。

在上面討論的四段程式，共有二種出口點。(1)和(2)只是顯示固定圖形，顯示完畢後便停止，等待使用者按下鍵盤上的任一鍵後才跳回 200 行的主迴圈（450及460 行之出口點）。第二種方式爲(3)和(4)所採用，在程式進行中只要按下鍵盤上任一鍵，便能使程式跳回主迴圈，故須在執行中不斷的測試鍵盤讀取位址，以決定是否有鍵被按下（700及990行的PEEK（E））。(3)採用這種方式是因爲萬花筒程式顯示的過程很長，沒有耐性看完的使用者可利用這種方式回到主迴圈。但當然顯示也有結束的時候，結束後程式會跳到450行的出口點，等使用者按下任一鍵後回到主迴圈。

```
]LIST

30 U = 9:D = 3:E =  - 16384:F =  -
     16368:G =  - 16287:H =  - 16
     286:Z = 127
100  POKE 768,173: POKE 769,48: POKE
     770,192: POKE 771,136: POKE
     772,208: POKE 773,4: POKE 77
     4,198: POKE 775,7
110  POKE 776,240: POKE 777,8: POKE
     778,202: POKE 779,208: POKE
     780,246: POKE 781,166: POKE
     782,6: POKE 783,76: POKE 784
     ,0: POKE 785,03: POKE 786,96
200  TEXT : HOME : PRINT : POKE F
     ,0
220 A$ = "APPLE DEMONSTRATION PRO
     GRAMS": GOSUB 2000: PRINT
230  PRINT "     TO OPERATE A DEMO
     NSTRATION,  TYPE"
240  PRINT "ITS NUMBER.  THEN TYP
     E THE KEY MARKED"
250  PRINT "'RETURN' AT THE RIGHT
      EDGE OF THE KEY-"
260  PRINT "BOARD.  TYPE THE 'RET
     URN' KEY TO STOP"
270  PRINT "ANY DEMONSTRATION.": PRINT
     : PRINT
290  PRINT "1.";
300 A$ = "STANDARD COLOR NAMES": GOSUB
     2000
310  PRINT "2.";
320 A$ = "STANDARD COLOR NUMBERS"
     : GOSUB 2000
330  PRINT "3.";
340 A$ = "KALEIDOSCOPE": GOSUB 20
     00
350  PRINT "4.";
360 A$ = "SKETCHING SCREEN": GOSUB
     2000
370  PRINT : PRINT
380  INPUT "WHICH WOULD YOU LIKE?
     ";A$
390  IF  LEN (A$) = 0 THEN  GOTO
     200
```

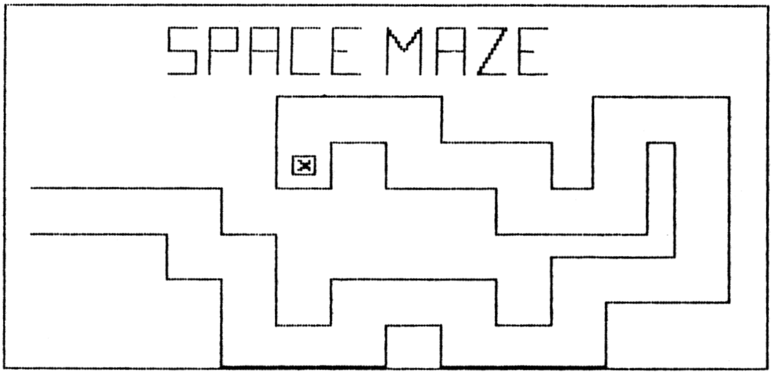
```
395 N =  ASC (A$) - 48
400  IF N < 1 OR N > 4 THEN  GOTO
     200
410  ON N GOTO 500,600,700,800
450  IF  PEEK (E) < Z THEN 450
460  GOTO 200
500  GOSUB 3000
510 A$ = "BMDPDGMLBOGPLYAW": GOSUB
     4000: PRINT
520 A$ = "LGBUGRBBRRRIGEQI": GOSUB
     4000: PRINT
530 A$ = "ATLRRELLWNENRLUT": GOSUB
     4000: PRINT
540 A$ = "KAUPNYUUNGYKNOAE": GOSUB
     4000
550  GOTO 450
600  GOSUB 3000
610  FOR I = 0 TO 14 STEP 2
620  HTAB 2 * I + 4: PRINT I;
630  NEXT I: PRINT
640  FOR I = 1 TO 15 STEP 2
650  HTAB 2 * I + 4: PRINT I;
660  NEXT I: PRINT : PRINT : GOTO
     450
700  GR : HOME : FOR W = 3 TO 50:
      FOR I = 1 TO 19: FOR J = 0 TO
     19:K = I + J: IF  PEEK (E) >
     Z THEN 200
750  COLOR= J * 3 / (I + 3) + I *
     W / 12
760  PLOT I,K: PLOT K,I: PLOT 40 -
     I,40 - K: PLOT 40 - K,40 - I
770  PLOT K,40 - I: PLOT 40 - I,K
     : PLOT I,40 - K: PLOT 40 - K
     ,I
780  NEXT J,I,W: GOTO 450
800  HOME :A$ = "USE THE GAME CON
     TROLS AND BUTTONS.": GOSUB 2
     000
810 A$ = "THE BUTTONS CLEAR THE T
     V AND CHANGE": GOSUB 2000
820 A$ = "THE DISPLAYED COLOR.  T
     RY ONE NOW.": GOSUB 2000
900  IF  PEEK (G) > Z OR  PEEK (H
     ) > Z THEN 930
```

```
910  IF  PEEK (E) > Z THEN 200
920  GOTO 900
930  HOME : GR
940 P =  PDL (0):X =  INT (P / 6)
     : IF X > 39 THEN X = 39
950 I =  PDL (1):Y =  INT (I / 6)
     : IF Y > 39 THEN Y = 39
960 P =  INT ((I + P) / 2 + 9): IF
     P > 255 THEN P = 255
970  GOSUB 1500
980  VTAB 21: PRINT "X IS ";X;" "
     : PRINT "Y IS ";Y;" "
990  IF  PEEK (E) > Z THEN 200
1000  IF  PEEK (G) > Z THEN  GR
1010  IF  PEEK (H) > Z THEN U = (
     U + 1)
1015  IF U = 16 THEN U = 0
1020  COLOR= U: PLOT X,Y: GOTO 94
     0
1500  POKE 6,P: POKE 7,D: CALL 76
     8: RETURN
2000  HTAB ((40 -  LEN (A$)) / 2 +
     1): PRINT A$: PRINT : RETURN

3000  GR : FOR I = 0 TO 31: COLOR=
     I / 2: VLIN 0,39 AT I + 3: NEXT
     I
3010  COLOR= 5: VLIN 0,39 AT 2: VLIN
     0,39 AT 35: HLIN 2,35 AT 0: HLIN
     2,35 AT 39: RETURN
4000  FOR I = 1 TO  LEN (A$)
4010  HTAB 2 * I + 2: PRINT  MID$
     (A$,I,1);
4020  NEXT I: RETURN

]
```

程式四　太空迷陣 (SPACE MAZE)



本程式的難易度僅屬於簡易但卻能產生很好的遊戲效果，非常適合我們在目前討論。本程式利用高解像畫面方式在螢幕上畫出一條彎彎曲曲的道路，道路的盡頭是一堆寶藏，而遊戲者的任務便是使用PADDLE控制一輛車子，沿著彎道到達寶藏處，取出寶藏。

這個遊戲雖然簡單，但設計的還不錯。高解像螢幕上除了顯示圖形之外，也顯示文字標題。遊戲本身，也分難、易兩種方式進行，由遊戲者在遊戲開始前選定。所謂程度較難者，就是遊戲

在進行中，遊戲者的車子常會受到一種磁力的影響而撞毀。除此之外，遊戲者車子的燃料有限，必須在限定的時間內到達寶藏處，否則還是無法達成任務。另外，遊戲者還可以選擇，車子在進行中是否要有聲音伴隨。而星際大戰的配樂更是本程式的一大特色。

本程式主要是提供我們一些遊戲程式的基本概念和製作方法，因此把它放在前面討論。

程式中主要變數說明

(1)（HV，VV）：爲遊戲者座標移動的垂直、水平方向之變量。

(2)（XO，YO）：爲遊戲者的前一個位置座標。

(3)（X，Y）：爲遊戲者的新位置座標。在一般遊戲中，繪出圖形進行的方式均爲：在新位置繪出圖形，再將前一個位置的圖形擦掉（即使用底色在原位置再畫一次圖形）。因此程式中使用（X，Y）及（XO，YO）分別記錄新的位置座標及前一個位置的座標。至於圖形移動的方式（如方向、速度），則完全由座標變量（HV，VV）所控制。如我們想描繪轟炸機所投炸彈的掉落圖形，則

須：

$$\begin{cases} HV = C \\ VV = VV + GRAVITY \end{cases} \Rightarrow \begin{matrix} X = X + HV \\ Y = Y + VV \end{matrix}$$

其中C，及GRAVITY均爲常數。

因炸彈移動，在水平方向爲水平等速進行，而在垂直方向則同受重力加速度影響爲等加速進行。故將HC設定爲常數代表每單位時間移動一定的距離（等速進行），而將VV = VV +（常數），代表在垂直方向因受定力的影響，使垂直方向的移動隨著時間的增加而愈來愈快。

而在本程式中，將變量的改變設定如下：

HV＝HV±1　⟹　X＝XO＋HV
VV＝VV±1　　　Y＝YO＋VV

代表在垂直方向及水平方向均受等加（或減）速度的影響。

(4) PT：記錄遊戲者在遊戲中所得之分數。

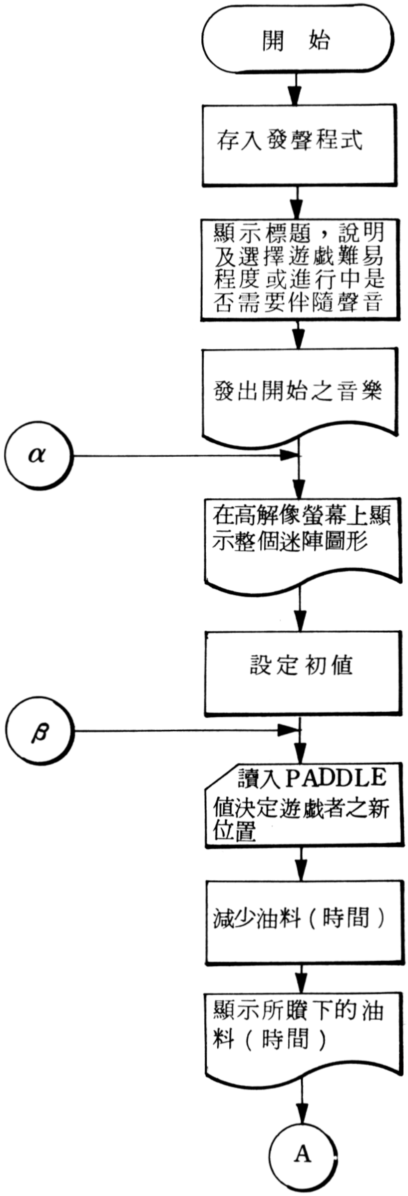
(5) TM：記錄遊戲者所剩下的油料，亦即控制遊戲者在遊戲中進行的時間，限制遊戲者必須在限定的時間內拿到寶藏。此外還有一個作用：當做遊戲者的得分。當遊戲者完成任務後，即以此所剩油料（時間）的多寡當做遊戲者得分的多寡。如此便可使遊戲者除了要得到寶藏外，還須以最短的時間到達，以增加遊戲的趣味性及緊張性。

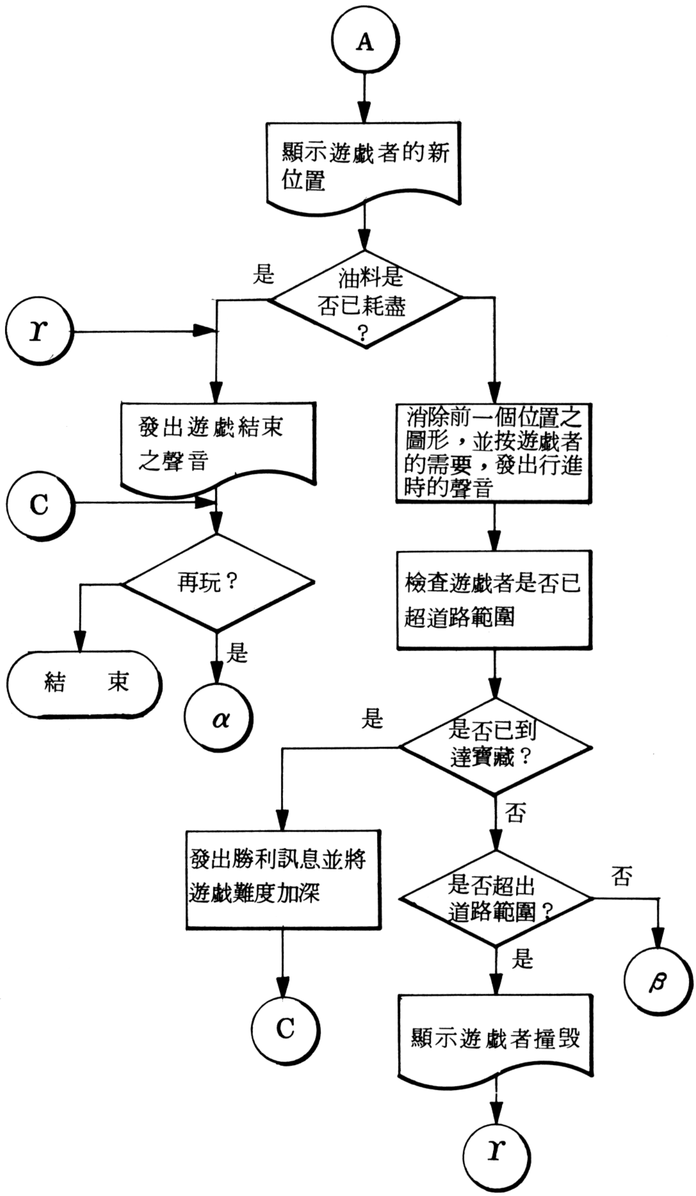
(6)（M1，M2）：M1為聲音的高低值，M2為聲音之長短，$0 \le M1，M2 \le 255$。

(7) Z：此變數用以知道遊戲者的車子是否已超出道路邊界而撞毀。程式中將全程道路路線分成 11 段各給予不同的值，若遊戲者的車子在這些道路中，則Z之值必為1至11 之值，否則便為0。故當Z＝0時便知道車子已超出道路而撞毀（後面詳細說明）。

程式之流程圖

以下將程式進行的主要流程以流程圖表示如下：（較詳細的說明請見下一部份“程式流程說明”）


開 始
存入發聲程式
顯示標題，說明及選擇遊戲難易程度或進行中是否需要伴隨聲音
發出開始之音樂
α
在高解像螢幕上顯示整個迷陣圖形
設定初值
β
讀入PADDLE值決定遊戲者之新位置
減少油料（時間）
顯示所賸下的油料（時間）
A


A
顯示遊戲者的新位置
油料是否已耗盡？
是
r
發出遊戲結束之聲音
C
再玩？
結　束
是
α
消除前一個位置之圖形，並按遊戲者的需要，發出行進時的聲音
檢查遊戲者是否已超道路範圍
是否已到達寶藏？
是
發出勝利訊息並將遊戲難度加深
C
否
是否超出道路範圍？
否
β
是
顯示遊戲者撞毀
r

程式流程說明

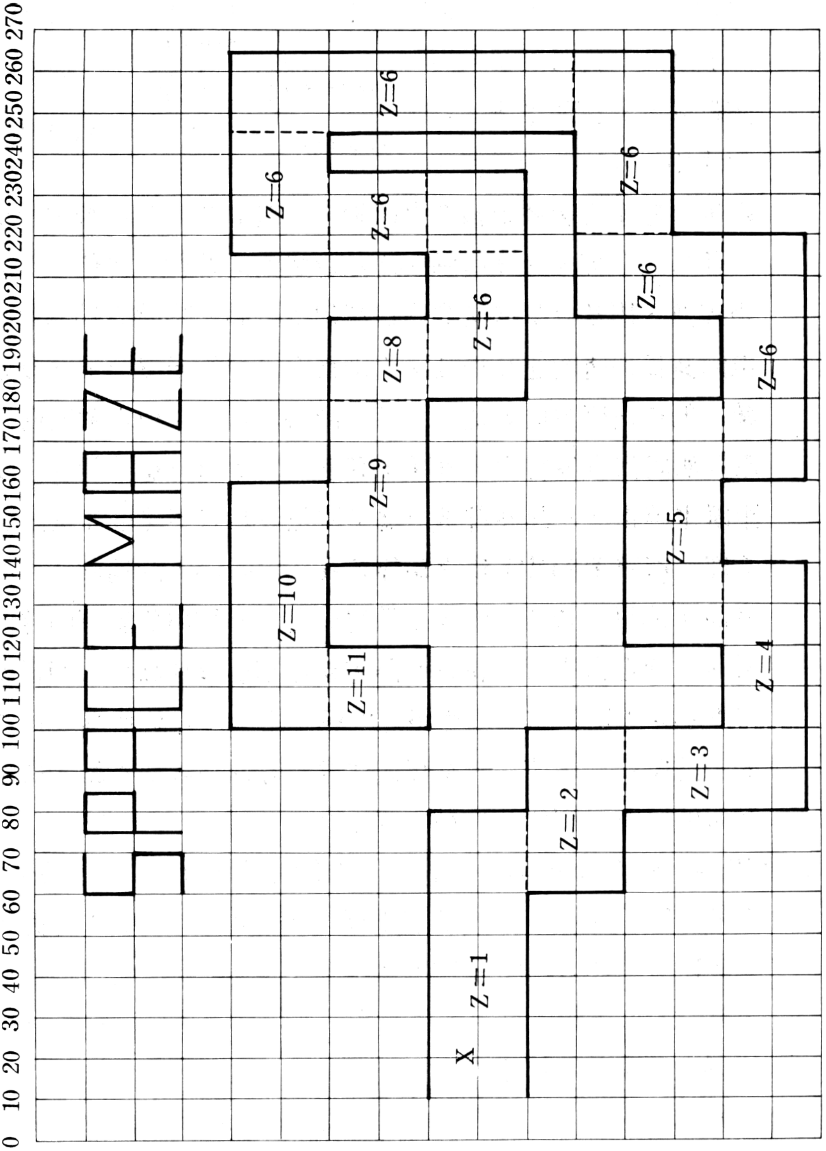
(1) 12-25 行 ：將機器語言發聲程式存入記憶體第三頁區。此發聲程式即為前面所介紹過的，只是儲存位址稍加改變。此發聲程式利用 $ 3 OC 存聲音的高低，$ 3 OD 存聲音的長短。

(2) 27-50 行：顯示說明，並請遊戲者選擇遊戲的難易度及是否需要音效。

(3) 97-175 行：檢查遊戲者的位置是否已超出道路範圍，或是否已到達寶藏區？其檢查方式為：將道路全程畫分為 11 段每段均由一數字代表（如下頁圖形），若遊戲者位於某段道路中，便將變數 Z 指定為該段數值，否則即為 0。最後只需測試 Z 之值即可知道是否已超出道路範圍或已到達寶藏區。由於道路曲折，若不用此法便很不容易測試遊戲者是否已超出道路範圍。

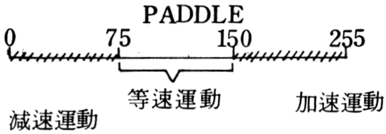
(4) 2000-2033 行：由 2000 行畫出圖形邊界； 2001-2003 繪出遊戲名稱"SPACE MAZE"; 2005-2033 行繪出迷陣圖形。圖形請見次頁）。

(5) 500-515 行： 使喇叭發出星際大戰的音樂。其方式為將聲音高低M1，長短M2分別存入 $ 30C，$ 30D 再呼叫發聲程式。詳細情形前面已討論過不再贅述。



SPACE MAZE
X
Z=1
Z=2
Z=3
Z=4
Z=5
Z=6
Z=8
Z=9
Z=10
Z=11
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160

(6) 200-242 行：此段主要在求出遊戲者新的位置座標。程式中使遊戲者可利用 PADDLE 加速或減速或維持等速度行進(如右圖），此外若遊戲者選擇難度較深的遊戲，則程式中用 RND 函數任意產生一磁場力影響遊戲者之運動。一般而言，若遊戲中利用這種方式（即加速度運動）控制遊戲者的移動，會使遊戲較為困難，因當遊戲者警覺到快超出道路範圍時雖立即減速，但因其為加速度運動仍無法立即轉向，只能慢慢減小移動距離（但仍向原方向前進），至一段時間後才能停止並轉向（原理如同汽車進行中，雖踩下剎車，汽車仍會繼續行走一段距離才停止）。此外遊戲中為了避免遊戲者不使用加減速度（即一直將 PADDLE 之值停留在 75 至 150 的範圍），故意設下油料（時間）的限制及得分之多寡按照所剩下的時間計算，使遊戲者若想在時限內完成任務或獲得高分，便需使用 PADDLE 加速（當然磁力的任意產生，也會使得遊戲者必須隨時注意減速）。本程式在這一方面設計的不錯，雖是短短的幾行，卻能發揮很大的作用。



(7) 243 -300 行：顯示剩下的油料及遊戲者目前的位置。程式中第 243 行 CALL -868 將游標所在的那一行，由底邊清除至游標的位置，因此 VTAB21：CALL-868：便能將螢幕上第 21 行清除，再顯示新的文字。這種方式在一般圖形文字混合幕中常常使用。

(8) 3000-3015 行：遊戲者已到寶藏區，遊戲再重新開始，但難度自動加深。

(9) 4000-4007 行：遊戲者超出道路範圍而撞毀，任務失敗。在

此程式中 4000-4005 行顯示車子撞毀的效果。

最後在結束本程式之前還有一點需要說明：即程式中 3000 行及 4000 行所使用的 POP 指令。程式中由 165 行的 GOTO 3000 進入 3000 行，但 165 行本身位於 100-175 行的副程式中，程式在進入此副程式前會先將 RETURN ADDRESS 存入系統堆疊區中（STACK 見第一章），故若直接由 165 行跳至 3000 行，而沒有執行 175 行的 RETURN 指令，會使此 RETURN ADDRESS 一直存在堆疊區中，造成錯誤故必須利用 POP 指令將此 RETURN ADDRESS 由堆疊區中清除。

```
]LIST

2  PRINT "";""
3  REM  **************************
4  REM  ** SPACE MAZE           **
5  REM  ** MICRO-SPAC           **
6  REM  ** P.O. BOX 325         **
7  REM  ** LINCOLN MASS 01773   **
8  REM  ** COPYRIGHT C 1980     **
9  REM  **************************
12  REM  THE FOLLOWING SUBROUTINE GOES
     INTO MEMORY BLOCK HEX $30E (782),
     THE TONE RESPOND TO POKE 0 TO 255
13  REM  PITCH=POKE 780,P   DURATION=PO
    KE 781,D
15  POKE 782,173: POKE 783,48: POKE 784
    ,192
16  POKE 785,136: POKE 786,208: POKE 78
    7,5
18  POKE 788,206: POKE 789,13: POKE 790
    ,3
20  POKE 791,240: POKE 792,9: POKE 793,
    202
22  POKE 794,208: POKE 795,245: POKE 79
    6,174
```

```
24  POKE 797,12: POKE 798,3: POKE
     799,76
25  POKE 800,14: POKE 801,3: POKE
     802,96
27  CALL  - 936
30  VTAB 7: INVERSE : HTAB 10: PRINT
     "** SPACE MAZE **": VTAB 23:
      HTAB 4: PRINT "COPYRIGHT 19
     80..MICRO-SPAC INC.": NORMAL
31  VTAB 9: PRINT "YOU WILL PILOT
      A SPACE CRUISER THRU THE": PRINT
     "STAR MAZE TO REACH THE PRIZ
     E DILITHIUM": PRINT "CRYSTAL
     S AT THE CENTER OF THE MAZE"
     : INVERSE : PRINT
32  PRINT "BE CAREFUL! IN THE FAR
     D VERSION OF THE": PRINT "GA
     ME YOUR SHIP IS PULL BUY HOS
     TIL  ": PRINT "MAGIC FORCE..
      SO TAKE CARE        "
33  PRINT "NOT TO CRASH!!!!!!!!!!
      !!!!!!!!!!!!!!!!!!"
34  NORMAL : PRINT "DO YOU WANT Y
     OUR SHIP SIGNAL SOUNDS?  ": INPUT
     "TYPE YES OR NO";NS$: INPUT
     "EASY OR HARD GAME? TYPE 'E'
      OR 'H'";HD$
38  GOSUB 500: PRINT "DO YOU WANT
      STARWARS MUSIC EACH GAME?":
      INPUT "Y OR N";M$
40  HGR : GOSUB 2000
45  IF M$ <  > "N" THEN  GOSUB 50
     0
50  GOTO 200
97  REM  THE FOLLOWING SUBROUTINE
      TESTS WETHER X AND Y ARE CO
     NTAINED IN THE SERIES OF 11
     RECTANGLES MAKING UP THE MAZ
     E
98  REM  IF X AND Y ARE SENSED, T
     HE Z IS SET THE NUMBER OF TH
     E RECTANGLE, AT THE END OF T
     HE TEST, Z IS TESTED, IF Z I
     S GREATER THAN
99  REM  ZERO IT MEANS X AND Y AR
     E IN BONDS.IF Z=0 THEN NO X
```

```
     AND Y HAVE BEEN SENSED IN BO
     UNDS AND THE PROGRAM GOES TO
      THE CRASH SUBROUTINE.
100  IF (X > = 10 AND X < = 80)
      AND (Y > = 80 AND Y < = 1
     00) THEN Z = 1
110  IF (X > = 60 AND X < = 100
     ) AND (Y > = 100 AND Y < =
     120) THEN Z = 2
120  IF (X > = 80 AND X < = 100
     ) AND (Y > = 120 AND Y < =
     158) THEN Z = 3
125  IF (X > = 100 AND X < = 14
     0) AND (Y > = 140 AND Y < =
     158) THEN Z = 4
130  IF (X > = 120 AND X < = 18
     0) AND (Y > = 120 AND Y < =
     140) THEN Z = 5
135  IF (X > = 160 AND X < = 22
     0) AND (Y > = 140 AND Y < =
     158) THEN Z = 6
137  IF (X > = 200 AND X < = 22
     0) AND (Y > = 110 AND Y < =
     140) THEN Z = 6
138  IF (X > = 220 AND X < = 26
     5) AND (Y > = 110 AND Y < =
     130) THEN Z = 6
139  IF (X > = 245 AND X < = 26
     5) AND (Y > = 40 AND Y < =
     110) THEN Z = 6
140  IF (X > = 215 AND X < = 24
     5) AND (Y > = 40 AND Y < =
     60) THEN Z = 6
141  IF (X > = 215 AND X < = 23
     5) AND (Y > = 60 AND Y < =
     100) THEN Z = 6
142  IF (X > = 180 AND X < = 23
     5) AND (Y > = 80 AND Y < =
     100) THEN Z = 6
145  IF (X > = 180 AND X < = 20
     0) AND (Y > = 60 AND Y < =
     100) THEN Z = 8
150  IF (X > = 140 AND X < = 18
     0) AND (Y > = 60 AND Y < =
     80) THEN Z = 9
```

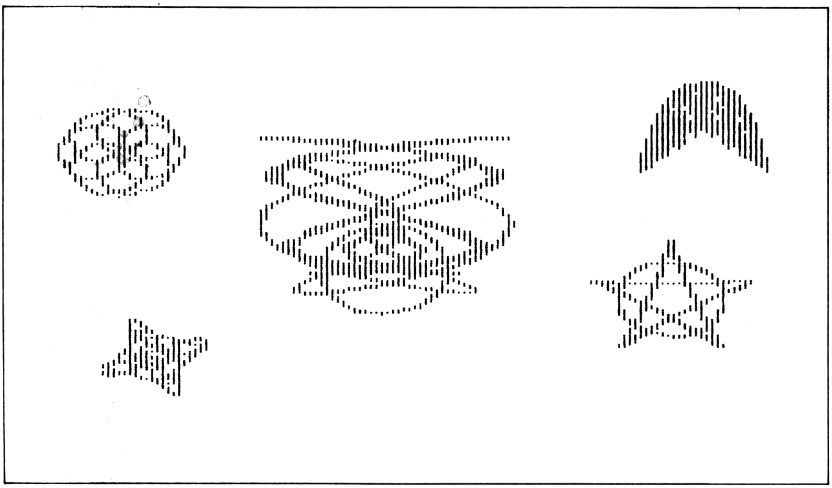
```
160  IF (X >  = 100 AND X <  = 16
     0) AND (Y >  = 40 AND Y <  =
     60) THEN Z = 10
162  IF (X >  = 100 AND X <  = 12
     0) AND (Y >  = 60 AND Y <  =
     80) THEN Z = 11
165  IF (X >  = 106 AND X <  = 11
     4) AND (Y >  = 66 AND Y <  =
     74) THEN 3000: REM  BRANCH T
     O WIN
170  IF Z <  = 0 THEN 4000: REM
     BRANCH TO CRASH...NO FLAGS W
     ERE SET TO INDICATE PRESENCE
      IN THE MAZE...THE REFORE MU
     ST BE OUTSIDE.
175 Z = 0: RETURN : REM  RESET Z
     EACH TEST
200 X = 15:Y = 90:HV = 0:VV = 0:T
     M = 600:XO = 15:YO = 90: CALL
      - 936
210  IF  PDL (0) >  = 150 THEN HV
      = HV + 1
220  IF  PDL (0) <  = 75 THEN HV =
     HV - 1
230  IF  PDL (1) >  = 150 THEN VV
      = VV + 1
231  IF HD$ = "E" THEN  GOTO 240
232  IF  RND (1) < .05 THEN HV =
     HV + 1
233  IF  RND (1) > .95 THEN VV =
     VV + 1
240  IF  PDL (1) <  = 75 THEN VV =
     VV - 1
242 X = XO + HV:Y = YO + VV
243 TM = TM - 1: VTAB 21: PRINT  TAB(
     10)"FUEL LEFT= ";TM: IF TM <
     100 THEN  VTAB 21: CALL  - 8
     68: PRINT  TAB( 10)"FUEL LEF
     T=";TM
245  VTAB 22: CALL  - 868: PRINT
     "HORIZ =";HV: PRINT  TAB( 25
     )"VERTICAL =";VV
260  HCOLOR= 3: HPLOT X,Y: IF PT =
     0 THEN 267
265  VTAB 23: PRINT  TAB( 4)"PREV
     IOUS RECORD SCORE IS: ":PT
```

```
267  IF TM <  = 0 THEN  CALL  - 9
     36: FLASH : PRINT  TAB( 10)"
     OUT OF FUEL";: PRINT  TAB( 1
     0)" ": GOTO 4005
270  IF X = XO AND Y = YO THEN 30
     0
280  HCOLOR= 0: HPLOT XO,YO: IF N
     S$ = "N" THEN 300
285  POKE 780,150: POKE 781,10: CALL
     782
300 XO = X:YO = Y: GOSUB 100: GOTO
     210
498  REM  THE 500 SUBROUTINE SETS
      UP THE MUSIC. 700 PLAYS IT
500 M1 = 255:M2 = 250: GOSUB 700:
     M1 = 170:M2 = 250: GOSUB 700
     :M1 = 190:M2 = 75: GOSUB 700
     :M1 = 203:M2 = 72: GOSUB 700
505 M1 = 230:M2 = 75: GOSUB 700:M
     1 = 126:M2 = 250: GOSUB 700:
     M1 = 170:M2 = 250: GOSUB 700
     :M1 = 190:M2 = 75: GOSUB 700
510 M1 = 203:M2 = 75: GOSUB 700:M
     1 = 230:M2 = 75: GOSUB 700:M
     1 = 126:M2 = 250: GOSUB 700:
     M1 = 170:M2 = 250: GOSUB 700
515 M1 = 190:M2 = 100: GOSUB 700:
     M1 = 203:M2 = 100: GOSUB 700
     :M1 = 190:M2 = 100: GOSUB 70
     0:M1 = 230:M2 = 250: GOSUB 7
     00: RETURN
700  POKE 780,M1: POKE 781,M2: CALL
     782: RETURN
2000  HCOLOR= 3: HPLOT 0,0 TO 279
     ,0 TO 279,159 TO 0,159 TO 0,
     0
2001  HPLOT 70,10 TO 60,10 TO 60,
     20 TO 70,20 TO 70,30 TO 60,3
     0: HPLOT 75,30 TO 75,10 TO 8
     5,10 TO 85,20 TO 75,20: HPLOT
     90,30 TO 90,10 TO 100,10 TO
     100,30: HPLOT 90,20 TO 100,2
     0
2002  HPLOT 115,10 TO 105,10 TO 1
     05,30 TO 115,30: HPLOT 130,1
     0 TO 120,10 TO 120,30 TO 130
```

```
       ,30: HPLOT 120,20 TO 125,20:
        HPLOT 140,30 TO 140,10 TO 1
       46,20 TO 152,10 TO 152,30
2003    HPLOT 158,30 TO 158,10 TO 1
       68,10 TO 168,30: HPLOT 158,2
       0 TO 168,20: HPLOT 173,10 TO
       183,10 TO 173,30 TO 183,30: HPLOT
       198,10 TO 188,10 TO 188,30 TO
       198,30: HPLOT 188,20 TO 193,
       20
2005    HPLOT 10,80 TO 80,80 TO 80,
       100 TO 100,100 TO 100,140 TO
       120,140 TO 120,120 TO 180,12
       0
2010    HPLOT 180,120 TO 180,140 TO
       200,140 TO 200,110 TO 245,11
       0 TO 245,60 TO 235,60 TO 235
       ,100 TO 180,100 TO 180,80 TO
       140,80 TO 140,60 TO 120,60 TO
       120,80 TO 100,80
2015    HPLOT 100,80 TO 100,40 TO 1
       60,40 TO 160,60 TO 200,60 TO
       200,80 TO 215,80 TO 215,40 TO
       265,40 TO 265,130 TO 220,130
        TO 220,158 TO 160,158
2020    HPLOT 220,158 TO 160,158 TO
       160,140 TO 140,140 TO 140,15
       8 TO 80,158 TO 80,120 TO 60,
       120 TO 60,100 TO 10,100
2030    HCOLOR= 3: HPLOT 106,66 TO
       114,66 TO 114,74 TO 106,74 TO
       106,66
2033    HPLOT 108,68 TO 112,72: HPLOT
       108,72 TO 112,68: RETURN
3000    POP : TEXT : FOR NN = 250 TO
       0 STEP  - 15: PRINT "** WINN
       ER **";: POKE 780,NN: POKE 7
       81,10
3005    CALL 782: NEXT NN: FOR N =
       1 TO 500: NEXT N: PRINT : PRINT
3010    IF TM > PT THEN  HOME : VTAB
       10: FLASH : PRINT "CONGRATUL
       ATIONS!": NORMAL : PRINT "YO
       U'VE BEATEN THE PREVIOUS HIG
       H SCORE ": PRINT "OF ";PT;"
       WITH YOUR SCORE OF ";TM
```

```
3011 GC = GC + 1: IF GC = 1 THEN
      PRINT : PRINT "IF YOU'VE PL
     AYING THE EASY GAME": PRINT
     "YOU ARE A WINNER!  NOW WE'L
     L ADVANCE TO": PRINT "THE HA
     RD GAME":HD$ = "H"
3012  IF TM > PT THEN PT = TM
3015  GOTO 4007
4000  POP : TEXT : FLASH : FOR NN
      = 1 TO 100: PRINT "** CRASH
      **";: NEXT NN: NORMAL
4005  FOR NN = 1 TO 250 STEP 50: POKE
     780,NN: POKE 781,50: CALL 78
     2: NEXT NN
4006  FOR NN = 1 TO 2000: NEXT NN
     : CALL  - 936
4007  INPUT "ANOTHER MISSION? (Y/
     N)";A$: IF A$ = "Y" THEN  GOTO
     40
4010  END
```

程式五　繪圖板程式
(APPLE PAINTBOX)



此程式主要是利用APPLE Ⅱ將電視螢幕（或TV Honitor）變成一個多功能繪圖板。使用者只須按下按鈕便可在螢幕上顯示直線或各種多邊形，或輸入幾個數值便可在螢幕上顯示美麗的圖樣。將此程式熟悉了後，更可創造出極優美的圖樣，是一個極有趣的繪圖程式。

除了程式本身的趣味性及實用性之外，程式的撰寫方式更是值得每一個想學習編寫應用性程式的讀者效法。此程式的寫法類似目前極流行的“MENU DRIVEN”的方式，並具有“ERRO-

R-PROOF"的能力。讀者在研習過本程式後，可修改此程式，使其更清楚、更方便使用，無形中更增加了自己的程式設計能力。

使用本程式除了APPLE之外，須使用搖桿（PADDLE），按鈕（PUSHBUTTON）及一部磁碟機。磁碟機的作用只是作圖形存取及讀取使用方法檔案內容之用，經過簡單的修改亦可省去磁碟機的功能。

使用方法說明

程式一開始，顯示"INSTRUCTION？"詢問使用者是否需要說明。按下"Y"後程式會顯示一連串的說明（見後）。說明部份顯示結束後按下"A"鍵可將說明再顯示一遍，否則按下＜RETURN＞便可正式開始。

正式開始後，螢幕上顯示黑色的高解像圖形文字混合幕，底下有四行文字說明的選擇項：

(1) COLOR：BLANK
(2) PEN：OFF
(3) PICTURE：PICTURE NOT NAMED
(4) SPECIAL EFFECTS （Q TO QUIF）

其中有一個選擇項以反白方式顯示。使用者可利用"→"，"←"鍵使各選擇項向下或向上輪流以反白方式顯示，至所要的選擇項後（即該選擇項以反白字形顯示）便可按下＜RETURN＞鍵進行下一步的動作。

①：選擇(1)後，螢幕底下會出現新的說明，並請使用者選擇一個顏色，然後再回到選擇項，但在COLOR之後會顯示使用者所選擇的顏色，而不再是BLANK。

②：選擇(2)後，程式自動轉換PEN之值由OFF至ON，或由ON至OFF。

③：選擇(3)後，程式會出現：

PICTURES：N（AME，L（OAD，S（AVE，
E（RASE，<ESC>

請使用者選擇所欲進行的動作。各符號意義如下：

N（AME：可對欲繪的圖形命名，或鍵入一已名命之圖形的名稱。

L（OAD：根據NAME，將此圖形由磁碟中存入記憶區中（$2000-$3FFF高解像圖形第一頁區）。

S（AVE：將螢幕上的高解像圖形，以NAME：後的名稱，當作檔案名稱，存入磁碟中。使讀者往後只須將N（AME設定爲此名稱，再選擇L（OAD便可將圖形再由磁碟中取出存入記憶體中。

E（RASE：將目前螢幕上的圖形清除。

<ESC>：回到程式一開頭的四個選擇項。

④：選擇(4)後，螢幕上會出現：

SPECIAL EFFECTS：F（ILL，C（URVES，
<ESC>

詢問使用者需要何種效果。選擇"C（URVES"，可在高解像螢幕上創造出很美麗的曲線圖案，這一部份我們留待後面再詳加說明。通常在選擇"F（ILL"之前，使用者應已在螢幕上設定四個定點，"FILL"的目的就是將此四點所圍成的四邊形用指定的COLOR塡滿。但事實上並不是如此簡單，四定點所指定的次序會影響FILL後的結果，使得結果不一定是塡滿顏色的四邊形也許是兩個三角形，關於這一點我們也留待稍後說明。此程式在設

計上有一個特點，即程式中繪圖是依固定的原則及次序進行，而使用者僅須改變其繪圖的次序便可創出十分美麗的圖形。

若使用者對四個選擇項已按本身的需要設定完畢，按下<ESC>便可進入圖形繪製程序（在繪圖進行中亦可隨時按下<ESC>回到設定選擇項的程序，即<ESC>用來選擇繪圖程序或設定選擇項的程序）。若在選擇項時設定PEN：ON則高解像螢幕會顯示出PADDLE 的移動軌跡，使使用者可將螢幕當成畫板，將 PADDLE當成畫筆，直接在螢幕上繪圖；若 PEN:OFF，則螢幕上便不會顯示出 PADDLE移動的軌跡，僅在PADDLE 最後的位置顯示一亮點，讓使用者知道目前的位置。PEN:OFF 的目的主要是讓使用者設定定點之用，此定點最多可設定四個。設定定點的作用是使SPECIAL EFFECTS中的FILL及CURVES 知道繪圖的範圍或中心。定點的設定是使用PDL #0，每按一下PDL #0程式便以 PADDLE目前在螢幕上的位置當作一定點；若使用者按的是PDL #1而不是PDL #0 則程式除了設定定點之外，並會從最新定點位置至前一個定點位置繪出一直線連接起來。這種方式可使我們很容易的在高解像螢幕上畫出各種斜率、長短、顏色不同的斜線；十分方便。

"使用方法說明"檔案的建立

一個設計完善的程式，必須對程式的使用方法有詳細的說明。本程式見於此，設計了一份十分詳細的使用方法說明，但若想在程式中利用 PRINT直接將這些說明文字顯示在螢幕上，則會使程式大部份都在做顯示的工作，並佔去許多的記憶位置；且使用者在看過數次之後，可能以後便很少會再去看，即程式的這些部份可能往後便很少被執行，而使得許多記憶區被浪費掉。因此

必須利用別的方式來進行這項工作。

本程式所採用的方式爲，將所有使用方法說明的文字建成一個 TEXT FILE（參考 DOS MANUAL）存入磁碟中，當使用者需要閱讀此使用方法說明時才從磁碟將此說明內容逐行從磁碟中取出，顯示在螢幕上。

建立此使用方法說明的TEXT FILE，與一般APPLESOFT BASIC及BINARY CODE FILE不同。讀者可利用一般TEXT EDITOR建立，不然亦可利用次頁所列出的TEXTFILE WRITER 建立此TEXT FILE，但須注意在所有說明文字內容建立完後，最後需再鍵入“＜FINSHED＞”，告訴此TEXTFILE WRITER 已建立完成。

使用方法說明檔案內容列在本節之後，空白部份均以逗點指示，便於讀者了解其格式。

使用方法說明檔案內容：

Listing 3 — PAINTBOX INSTRUCTIONS File
(Blanks replaced with periods for easy unformatted entry)

```
........................................
1. PRESS <ESC> TO GET TO AND FROM MENU.
...SELECT A MENU ITEM WITH THE ARROW
...KEYS. PRESS <RETURN> TO CARRY OUT
...THE ACTION SELECTED.
........................................
2. SELECT THE ITEM "COLOR:" AND PRESS
...<RETURN>. CHOOSE A COLOR.
........................................
3. DRAWING WITH PEN:OFF
........................................
...PRESS <ESC> TO LEAVE THE MENU. THE
...PADDLE CONTROLS [PDL(0) AND PDL(1)]
...MOVE A SPOT ON THE SCREEN. BUTTON
...0 MAKES A MARK; BUTTON 1 DRAWS A
...LINE TO THAT MARK IN THE CHOSEN
...COLOR. THE OLD ENDING POINT BECOMES
...THE NEW STARTING POINT.
```

```
..........................................
..........................................
..........................................
4. DRAWING WITH PEN:ON
..........................................
...PUSH BUTTON 0 TO MAKE A MARK ON THE
...SCREEN, THEN PRESS <ESC> TO GET TO
...THE MENU. SELECT THE ITEM "PEN:"
...WITH THE ARROW KEYS AND PRESS
...<RETURN>. THE PADDLE CONTROLS NOW
...WORK LIKE THE KNOBS ON AN
..."ETCH- A-SKETCH".
..........................................
5. SPECIAL EFFECTS: F(ILL
..........................................
...FILL IS USED TO FILL AREAS WITH A
...SOLID COLOR. USE BUTTON 0 TO SET 4
...POINTS IN THE FOLLOWING ORDER:
.............1..........2
..............*........*
...........................................
...........................................
...........................................
............*...........*
...........3.............4
...THEN PRESS <ESC> TO GET THE MENU,
...SELECT "SPECIAL EFFECTS", AND
...PRESS "F" FOR FILL. THE AREA INSIDE
...THE 4 POINTS WILL BE FILLED WITH THE
...SELECTED COLOR. OF COURSE, THE
...POINTS CAN BE ENTERED IN OTHER
...ORDERS, WHEREUPON OTHER THINGS WILL
...HAPPEN. TRY IT.
..........................................
6. SPECIAL EFFECTS: C(URVES
..........................................
...THIS SPECIAL EFFECT WORKS SOMEWHAT
...LIKE A SPIROGRAPH. MARK 2 POINTS
...WITH BUTTON 0 IN THIS FORMAT:
...........................................
.................. 1
...................*
...........................................
...........................................
...................*
..................2
...........................................
...THESE POINTS DETERMINE THE SIZE AND
...LOCATION OF THE CURVE. PRESS <ESC>
...TO GET THE MENU, CHOOSE "SPECIAL
...EFFECTS", AND PRESS "C" FOR CURVES.
...ENTER VALUES FOR A AND B (THESE
...DETERMINE THE SHAPE OF THE CURVE).
...NEGATIVE NUMBERS INVERT THE CURVE.
```

```
...THE CURVE "BOUNCES OFF" THE TOP,
...BOTTOM, & SIDES OF SCREEN. NEXT
...ANSWER "MODULATE THE CURVE?" WITH Y
...OR N. MODULATION ALTERS THE BASIC
...SHAPE. ENTER NUMBERS FOR "% AMPLI-
...TUDE" AND "FREQUENCY". USING 3,3 FOR
...A AND B (WHICH, INCIDENTALLY,
...PRODUCES A CIRCLE), TRY:
......................................
.......AMPLITUDE=20,FREQUENCY=9
.......AMPLITUDE=30,FREQUENCY=9
.......AMPLITUDE=30,FREQUENCY=-9
......................................
......................................
......................................
......................................
......................................
...OTHER INTERESTING SHAPES ARE
...PRODUCED BY A=4,  B=4;
................A=10, B=23;
................A=33, B=33;
...........AND A=12, B=6.
......................................
...FINALLY, TRY A=-3,B=-3,AMPLITUDE=30,
...FREQUENCY=3. SURPRISE!!
......................................
................**NOTE**
......................................
.BUTTON 1 CAN BE USED TO BREAK OUT OF
......EITHER EFFECT AT ANY TIME.
.IN CASE OF ACCIDENTAL EXIT, RETURN TO
..BASIC BY ENTERING 3D0G <RETURN> IF
REQUIRED, THEN TYPE GOTO 1000 <RETURN>.
......................................
......................................
```

程式：TEXTFILE WRITER

```
10  REM TEXTFILE WRITER
20  D$=CHR(4):I$=" "
30  F$="PAINTBOX INSTRUCTIONS":
    REM NAME OF FILE TO BE WRITTEN
50  PRINT D$;"OPEN";F$:PRINT D$;
    "DELETE";F$: PRINT D$;"OPEN"
```

```
          ; F$
60    GET CH$ : PRINT CH$
65    IF CH$=CHR$(21) AND I$< >""
      THEN I$ = LEFT$( I$,LEN( I$)-1):
      POKE 36, POS(0) - 1
70    IF CH$<> CHR$(13) THEN
      I$ = I$ + CH$ : GOTO 60
75    IF I$ = "<FINISHED>" THEN 100
80    PRINT D$:PRINT D$;"APPEND";F$:
      PRINT D$;"WRITE";F$:PRINT I$:
      I$ = "" : GO TO 60
100   PRINT:PRINT D$;"CLOSE";F$
105   PRINT"FINISHED":END
```

至於 PAINTBOX INSTRUCTIONS 之檔案內容請見前頁。

程式使用變數說明

(1) SW，SW$：用以指示 PEN：ON 或 OFF， ON 時 SW=1，OFF 時 SW=0；SW$ 供顯示用故直接存"ON"或"OFF"。

(2) ML，MH：因本程式將 HIMEM 設定為 8192，而將原來的 HIMEM 之值存入此變數中，待程式結束前再利用此值將 HIMEM 還原。

(3) PIC$：用以儲存使用者所設定的圖形名稱。字串長度為 25 個字元。若圖形須 LOAD 或 SAVE 時均使用"PBX" 加上此字串變數內容當作檔案名稱，但由於 APPLE 規定各檔案

名稱不得超過 30 個字元，故將此字串長度限定為 25 個字元長度。

(4) COL，COL$：COL 儲存使用者所指定的顏色碼；COL$ 則儲存該顏色的英文名稱，供螢幕顯示之用。

(5) H（4），V（4）：此陣列變數用以儲存使用者所設定的四個定點的 X 座標及 Y 座標。

(6)（XO，YO）：用以儲存使用者最新設定的定點座標。

(7)（X1，Y1）：為 PADDLE 值對應到螢幕的最新位置座標。

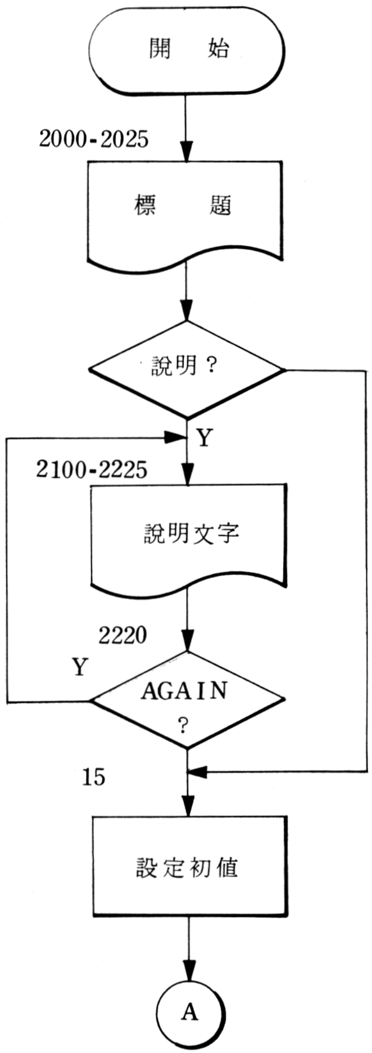
(8)（X2，Y2）：為前一次的 PADDLE 值所對應的位置座標。

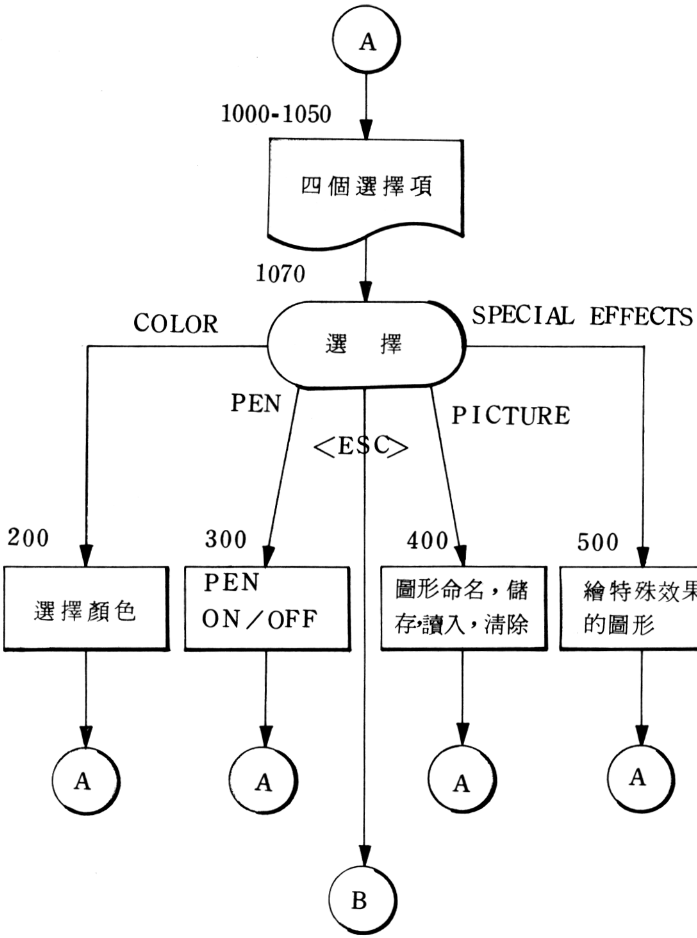
(9)（X3，Y3）：為用 PUSHBUTION#1 設定定點時，該定點的座標，因若使用 PUSHBUTTON#1 設定定點時，程式會從此定點至前一次定點用色線連接繪出直線，故先暫時用此變數儲存最新定點座標，等直線繪完後，再將（X3，Y3）的內容存至（X0，Y0）。

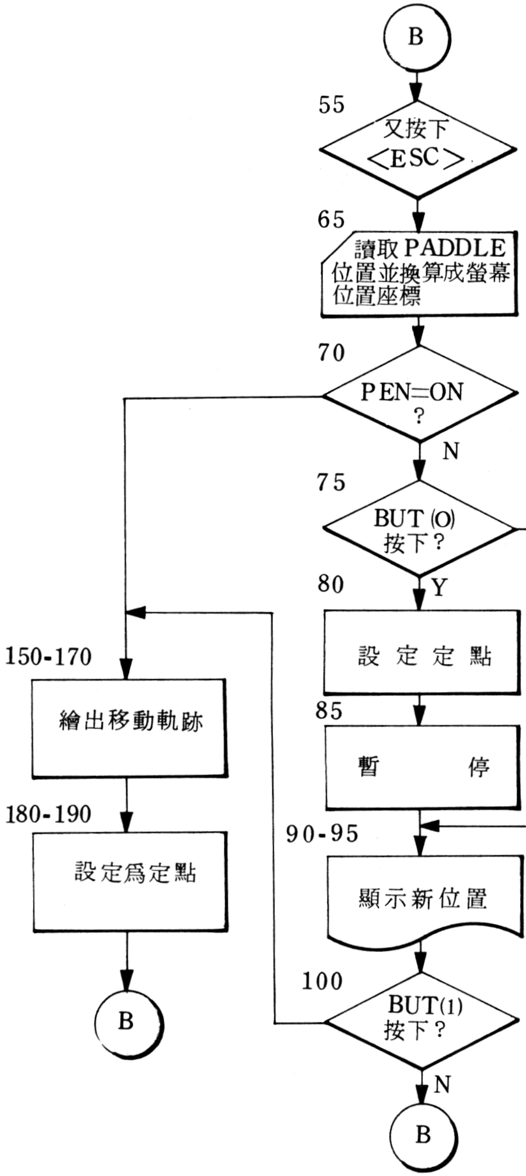
除了以上幾個變數程式中還使用到幾個常數，留待程式部份再說明。

程式流程說明

首先利用流程圖將程式的整個流程表示如下：

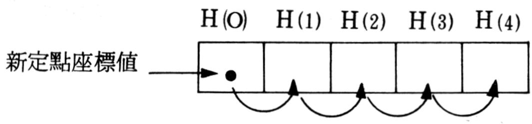

開 始
2000-2025
標 題
說明？
Y
2100-2225
說明文字
2220
Y
AGAIN ?
15
設定初值
A


A
1000-1050
四個選擇項
1070
選　擇
COLOR
SPECIAL EFFECTS
PEN
PICTURE
<ESC>
200
300
400
500
選擇顏色
PEN ON／OFF
圖形命名，儲存，讀入，清除
繪特殊效果的圖形
A
A
A
A
B


B
55
又按下
<ESC>
65
讀取PADDLE
位置並換算成螢幕
位置座標
70
PEN=ON
?
N
75
BUT (O)
按下？
N
80
Y
設定定點
85
暫停
90-95
顯示新位置
100
BUT(1)
按下？
N
B
150-170
繪出移動軌跡
180-190
設定爲定點
B

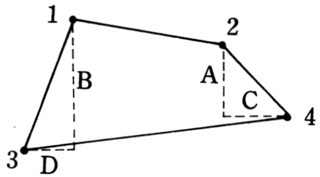
上面已將整個程式流程表示的很清楚了。下面只對一些須要說明的幾個部份說明，其他部份請參考上述流程圖及前面的說明。

(1) 150-190 行：此段副程式有二個進入點，PEN：ON時由150行爲進入點；PEN：OFF時則以 180 行爲進入點。150行至 170行主要是繪出移動軌跡，180-190 行則爲定點設定，其方式爲，將H，V陣列當做儲存定點座標之用。每次要設定一新定點時，便將此定點座標存入H（0），V（0），然後向前進一個位置，如下圖：



如此便可將新的定點存入H（1），V（1）中，原來H（1），V（1）亦前進到H（2），V（2），…… 以此類推，故最多可儲存四個定點。

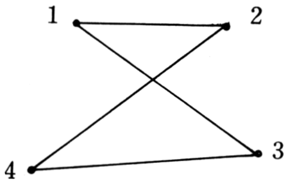
(2) 705-805 行：此段程式的目的爲將四定點所圍成的區域填滿顏色，但因其有一定之繪圖次序故若定點次序不同常能產生別的效果。程式中假設各定點的形成次序及在螢幕上的次序爲：

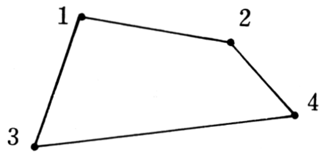


程式中 705-720 行計算A，B，C，D 之數值，若爲 0 則假設爲 1×10^{-6}，避免產生 0 爲分母的情形。程式 735 行C／A爲點 2，4 連線的斜率。X 9＝H（4）＋N＊C／A，其中N由 0 漸增至A，便可使X 9 由H（2）逐漸向H（4）靠

近；750行的Y9＝V（4）＋N，便可使Y9由V（2）逐漸向V（4）靠近，如此即（X9，Y9）由定點2沿著點2、4連線上漸漸向定點4靠近最後到達點4。（X8，Y8）的情形與（X9，Y9）類似，由定點1沿著點1、3連線向點3靠近，但由於必須使Y8改變的次數與Y9相同（才能使用HPLOT X9，Y9 TO X8，Y8正好將四邊形填滿），故程式780行計算Y8＝V（3）＋N＊B／A。

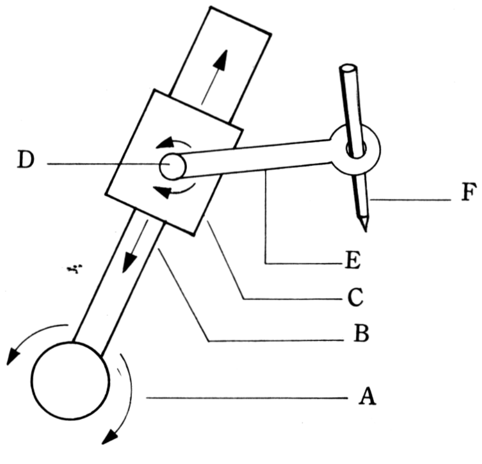
若我們將定點次序以下圖方式設定，則其結果：





其他還有許多種情形，讀者可在螢幕上試試看。

(3) 535-650行：此部份幾乎無法對其做完整的描述，下圖中我們將此部份程式假想爲一機械裝置，其動作的近似方式約爲：



主臂B可繞著主軸A旋轉，移動台架C可在主臂B旋轉時在主臂B上前後移動。同時支臂E（其長度即程式中的“%AMPLITUDE”）繞著軸D旋轉，其旋轉率則由程式中的“FREQUENCY”控制，F為畫筆執行眞正的畫圖動作。

程式570-645行以極座標方式進行此繪圖動作。且605-620行檢查點座標是否超過螢幕範圍，若已超過則以該點對上、下、左、右螢幕邊界“反彈”後的位置當做畫點位置。在繪圖進行時，可隨時按下PUSHBOTTON # 1 結束繪圖動作。

此外程式中使用到幾用常數：

① P2之值即為 $2\pi = 6.28318531$

② T＝3使SIN函數的週期由 2π 減為 $\frac{2\pi}{3}$

③ S＝0.03498066585 換成圓周度數約為2°，故570行的TH便可由0°每次增加2°直到360°為止。

④ K＝1.18則用以放大X座標值。

其他的A，B，M，N則可使得極座標方程式改變而產生各種不同的極座標圖案（也因此使得我們無法分析此極座標方程式，只能以上面機械圖形說明其約略動作）。

```
]LIST

4  PRINT ""
10  GOTO 2000
15 SW$ = "OFF":ML =  PEEK (115):M
     H =  PEEK (116): HIMEM: 8192
20 COL = 0:COL$ = "BLAKE      "
25 PIC$ = "PICTURE NOT NAMED
         "
```

```
30 KBD =  - 16384:SB =  - 16368:B
     1 =  - 16287:B2 =  - 16286:T
      = 3:K = 1.18
35 P2 = 6.28318531:S = .034980665
     85: DIM H(4),V(4):D$ =  CHR$
     (4)
40 X0 = 5:Y0 = 5:COL = 0:COL$ = "
     BLACK      ":PIC$ = "PICTURE
     NOT NAMED      "
45  HGR : GOTO 1000
50  REM  BASIC DRAW LOOP
55  IF  PEEK (KBD) = 155 THEN  POKE
     SB,0: GOTO 1000
60 X2 = X1:Y2 = Y1
65 X1 =  INT ( PDL (0) * 277 / 25
     5 + 1):Y1 =  INT ( PDL (1) *
     189 / 255 + 1)
70  IF SW THEN  GOSUB 150: GOTO 5
     0
75  IF  PEEK (B1) <  = 127 THEN 9
     0
80 X0 = X1 + 1:Y0 = Y1 + 1: HCOLOR=
     3: HPLOT X0,Y0: GOSUB 175
85  FOR D = 1 TO 500: NEXT D
90  HCOLOR= 3: HPLOT X1,Y1
95  HCOLOR= 0: HPLOT X2,Y2
100  IF  PEEK (B2) > 127 THEN  GOSUB
      150
105  GOTO 50
150  REM  PLOT AND UPDATE
155 X3 = X1:Y3 = Y1
160  HCOLOR= COL
165  HPLOT X0,Y0 TO X3,Y3: HPLOT
      X0 - 1,Y0 TO X3 - 1,Y3
170 X0 = X3:Y0 = Y3
175  REM  ENTER HERE TO UPDATE ON
      LY
180 H(0) = X0:V(0) = Y0
185  FOR I = 4 TO 1 STEP  - 1:H(I
      ) = H(I - 1):V(I) = V(I - 1)
      : NEXT
190  RETURN
200  REM  SELECT COLOR
205  HOME
210  PRINT "COLOR: B(LACK, L(IGHT
```

```
     BLUE, W(HITE,": PRINT "
        O(RANGE, P(URPLE, G(REEN
    "
215  GET CH$
220  IF CH$ = "B" THEN COL$ = "BL
     ACK      ":COL = 0: GOTO 1000

225  IF CH$ = "L" THEN COL$ = "LI
     GHT BLUE":COL = 6: GOTO 1000

230  IF CH$ = "W" THEN COL$ = "WH
     ITE      ":COL = 3: GOTO 1000

235  IF CH$ = "O" THEN COL$ = "OR
     ANGE     ":COL = 5: GOTO 1000

240  IF CH$ = "P" THEN COL$ = "PU
     RPLE     ":COL = 2: GOTO 1000

245  IF CH$ = "G" THEN COL$ = "GR
     EEN      ":COL = 1: GOTO 1000
250  GOTO 205
300  REM  TURN PEN ON OR OFF
305 SW =  NOT SW
310  IF  NOT SW THEN SW$ = "OFF"
315  IF SW THEN SW$ = " ON"
320  GOTO 1000
400  REM  CHANGE PICTURE
404  HOME
408  PRINT "PICTURE: N(AME, L(OAD
     , S(AVE,": PRINT "          E
     (RASE, <ESC>"
412  ONERR  GOTO 484
416  GET CH$
420  IF CH$ =  CHR$ (27) THEN  POKE
     216,0: GOTO 1000
424  IF CH$ = "E" THEN  HOME : PRINT
     "ARE YOU SURE? ";: GET CH$: IF
     CH$ = "Y" THEN PIC$ = "PICTU
     RE NOT NAMED         ": HGR :
      POKE 216,0: GOTO 1000
428  IF CH$ = "N" THEN 460
432  IF CH$ = "S" THEN 476
436  IF CH$ <  > "L" THEN  POKE 2
     16,0: GOTO 404
```

```
440  HOME : PRINT "NAME OF PICTUR
     E TO LOAD:"
444  INPUT "";PIC$: IF  LEN (PIC$
     ) > 25 THEN PIC$ =  LEFT$ (P
     IC$,25)
448  IF  LEN (PIC$) < 25 THEN PIC
     $ = PIC$ + " ": GOTO 448
452  PRINT D$;"BLOAD PBX.";PIC$;"
     ,A$2000"
456  POKE 216,0: GOTO 1000
460  HOME : PRINT "NAME THIS PICT
     URE:"
464  INPUT "";PIC$: IF  LEN (PIC$
     ) > 25 THEN PIC$ =  LEFT$ (P
     IC$,25)
468  IF  LEN (PIC$) < 25 THEN PIC
     $ = PIC$ + " ": GOTO 468
472  POKE 216,0: GOTO 404
476  PRINT : PRINT D$;"BSAVE PBX.
     ";PIC$;",A$2000,L$1FFF"
480  POKE 216,0: GOTO 1000
484  REM  LOAD/SAVE ERRORS
488 EC =  PEEK (222): IF EC = 6 THEN
      PRINT "NOT ON THIS DISK": FOR
     D = 1 TO 1500: NEXT : POKE 2
     16,0: GOTO 404
492  IF EC = 9 THEN  PRINT D$;"DE
     LETE PBX.";PIC$: PRINT "THIS
      DISK IS FULL": FOR D = 1 TO
     2500: NEXT : POKE 216,0: GOTO
     404
496  PRINT "ERROR ENCOUNTERED. CH
     ECK DISK, DRIVE,   SYNTAX AN
     D TRY AGAIN.": FOR D = 1 TO
     2500: NEXT : POKE 216,0: GOTO
     404
500  REM  SPECIAL EFFECT
505  HOME
510  PRINT "SPECIAL EFFECT: F(ILL
     ,C(URVES,<ESC>"
515  GET CH$
520  IF CH$ =  CHR$ (27) THEN 100
     0
525  IF CH$ = "F" THEN 700
530  IF CH$ <  > "C" THEN 505
```

```
535 Z =  ABS (V(2) - V(1))
540  HOME : INPUT "ENTER NUMBER F
     OR 'A' AND <RETURN>: ";A
545  INPUT "ENTER NUMBER FOR 'B'
     AND <RETURN>: ";B
550  PRINT "MODRULATE THIS CURVE?
      ";: GET CH$: PRINT : IF CH$
      <  > "Y" THEN M = 0:N = 0: GOTO
     565
555  INPUT "ENTER % AMPLITUDE AND
      <RETURN> :";M:M = M / 100 *
     Z
560  INPUT "ENTER FREQUENCY AND <
     RETURN>: ";N
565  HCOLOR= COL
570  FOR TH = 0 TO P2 STEP S
575 R = Z *  SIN (TH * T)
580 X2 = K * R *  COS (A * TH) +
     H(2)
585 X2 =  ABS (X2 + (K * M *  COS
     (N * TH * T)))
591 Y2 = R *  SIN (B * TH) + V(2)

595 Y2 =  ABS (Y2 + (M *  SIN (N *
     TH * T)))
600  IF TH = 0 THEN X1 = X2:Y1 =
     Y2
605  IF X1 > 278 THEN X1 = 278 -
     (X1 - 278)
610  IF Y1 > 190 THEN Y1 = 190 -
     (Y1 - 190)
615  IF X2 > 278 THEN X2 = 278 -
     (X2 - 278)
620  IF Y2 > 190 THEN Y2 = 190 -
     (Y2 - 190)
625  HPLOT X1,Y1 TO X2,Y2
630  HPLOT X1 + 1,Y1 TO X2 + 1,Y2
635 X1 = X2:Y1 = Y2
640  IF  PEEK (B2) > 127 THEN 100
     0
643  NEXT TH
650  GOTO 1000
700  REM  FILL
705 A = V(2) - V(4): IF A = 0 THEN
     A = 1E - 6
```

```
710 B = V(1) - V(3): IF B = 0 THEN
     B = 1E - 6
715 C = H(2) - H(4): IF C = 0 THEN
     C = 1E - 6
720 D = H(1) - H(3): IF D = 0 THEN
     D = 1E - 6
725  HCOLOR= COL
730  FOR N = 0 TO A
735 X9 = H(4) + N * C / A
740  IF X9 < 0 THEN X9 = 0
745  IF X9 > 279 THEN X9 = 279
750 Y9 = V(4) + N
755  IF Y9 < 0 THEN Y9 = 0
760  IF Y9 > 191 THEN Y9 = 191
765 X8 = H(3) + N * D / B
770  IF X8 < 0 THEN X8 = 0
775  IF X8 > 279 THEN X8 = 279
780 Y8 = V(3) + N * B / A
785  IF Y8 < 0 THEN Y8 = 0
790  IF Y8 > 191 THEN Y8 = 191
795  HPLOT X9,Y9 TO X8,Y8
800  IF  PEEK (82) > 127 THEN 100
     0
805  NEXT N
810  GOTO 1000
1000  REM  NEW PAINTBOX MENU
1005 F = 1
1010  POKE  - 16304,0: POKE  - 16
     297,0: POKE  - 16301,0: POKE
     34,20: HOME
1015  IF F = 1 THEN  INVERSE
1020  HTAB 3: PRINT "COLOR: ";COL
     $;"                    ": NORMAL

1025  IF F = 2 THEN  INVERSE
1030  HTAB 3: PRINT "PEN: ";SW$;"
                       ": NORMAL
1035  IF F = 3 THEN  INVERSE
1040  HTAB 3: PRINT "PICTURE: ";P
     IC$: NORMAL
1045  IF F = 4 THEN  INVERSE
1050  HTAB 3: PRINT "SPECIAL EFFE
     CTS        (Q TO QUIT) ";: NORMAL

1055  GET CH$
```

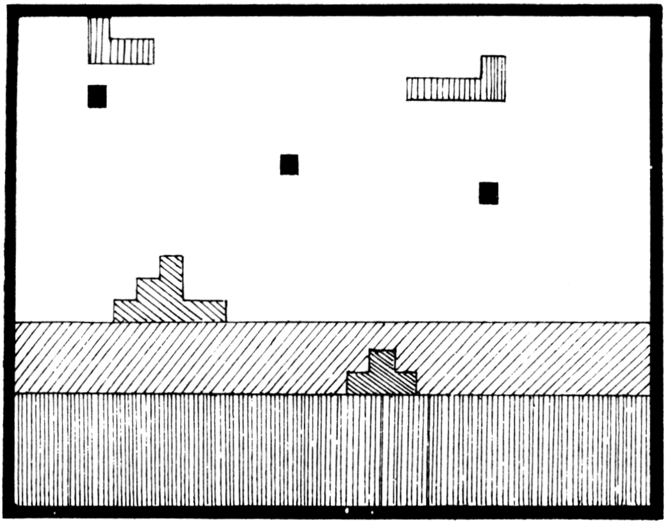
```
1060  IF CH$ =  CHR$ (21) THEN F =
     F + 1: IF F > 4 THEN F = F -
     4
1065  IF CH$ =  CHR$ (8) THEN F =
     F - 1: IF F < 1 THEN F = F +
     4
1070  IF CH$ =  CHR$ (13) THEN  ON
     F GOTO 200,300,400,500
1075  IF CH$ =  CHR$ (27) THEN  POKE
      - 16302,0: GOTO 50
1080  IF CH$ = "Q" THEN  POKE 115
     ,ML: POKE 116,MH: POKE 34,0:
      TEXT : POKE  - 16298,0: HOME
     : END
1085  GOTO 1010
2000  REM  TITLE AND INSTRUCTIONS

2005  HOME
2010  VTAB 11: HTAB 13: INVERSE :
      PRINT "---PAINTBOX---": NORMAL
2015  PRINT : HTAB 8: PRINT "COPY
     RIGHT 1980 BY AL EVANS"
2025  VTAB 24: HTAB 18: PRINT "IN
     STRUCTIONS?";: GET CH$: IF C
     H$ = "Y" THEN 2100
2030  CLEAR : GOTO 15
2100  REM  GET INSTRUCTIONS FORM
     DISK
2105 LC = 0:I$ = "":D$ =  CHR$ (4
     ): HOME
2110  ONERR  GOTO 2200
2115  PRINT : PRINT D$;"OPEN PAIN
     TBOX INSTRUCTIONS"
2120  PRINT : PRINT D$;"READ PAIN
     TBOX INSTRUCTIONS"
2125  GET CH$: IF CH$ <  >  CHR$
     (13) THEN I$ = I$ + CH$: GOTO
     2125
2130  PRINT : PRINT I$:I$ = "":LC
      = LC + 1: IF LC < 22 THEN 2
     125
2135  PRINT : PRINT D$: HTAB 11: INVERSE
     : PRINT "<RETURN> FOR MORE";: NORMAL
2140  GET CH$: IF CH$ <  >  CHR$
     (13) THEN 2140
```

```
2145 LC = 0: GOTO 2120
2200  PRINT : VTAB 24: HTAB 15: PRINT
     "**THE END**"
2205  PRINT "PRESS 'A' TO SEE INS
     TRUCTIONS AGAIN"
2210  PRINT "PRESS <RETURN> TO ST
     ART PAINTING"
2215  PRINT D$;"CLOSE PAINTBOX IN
     STRUCTIONS"
2220  GET CH$: IF CH$ = "A" THEN
     2100
2225  POKE 216,0: CLEAR : GOTO 15


]
```

程式六　海空大戰
(AIRSEA BATTLE)



本程式用 INTEGER BASIC 所寫成，用以增加程式的執行速度。因此對一些不會 INTEGER BASIC 的讀者可能會產生困擾，不過不需要擔心，何不趁此機會使您的 INTEGER BASIC 速成呢！只要翻開書後的附錄一，花上半個鐘頭您就可知道如何使用 INTEGER BASIC 及它與 APPLESOFT 的差別了。甚至讀者可花些時間將程式轉換成 APPLESOFT 的海空大戰程式（如果您沒有 INTEGER BASIC 編譯器），只是速度上可能會打些折扣。

程式中使用低解像螢幕做爲圖形的顯示，且爲了加快圖形的顯示，亦使用前面討論過的低解像圖形表顯示圖形。程式中使用到許多變數，且螢幕上顯示的圖形很多又因將螢幕劃分爲天空及大海增加圖形處理上的困難。總而言之，本程式包括了大部份這類型遊戲程式的特色，是很值得我們討論的程式。

遊戲說明

海空大戰，顧名思義，可以知道在遊戲中將敵對雙方劃分爲空軍和海軍兩方面進行比賽。可由二人對玩或一人與機器比賽，遊戲者可選擇加入海、空軍任何一方。空軍擁有戰鬥機及轟炸機可同時出現在螢幕上，海軍則擁有潛艇與驅逐艦亦可同時出現在螢幕上或個別出現。空軍可對海軍船隻投射炸彈予以摧毀，海軍亦可對空軍發射飛彈，但雙方在一回合的遊戲中，最多只能丟下（或發射） 15 枚炸彈，且一回合的遊戲也有時間的限制，時間一到遊戲自動結束，並按照各方所擊落（或擊沈）對方軍備之種類及數目顯示兩方面的勝負、得分並評定兩方面在此次戰役中的表現。

以上是整個遊戲的大概，此外還有一些細節說明如下。海、空軍雙方的飛機或船隻在螢幕上是以不定時間間隔及不定速度出現（戰鬥機及驅逐船由左向右行進，潛艇及轟炸機由右向左），以避免敵方的預先瞄準。戰鬥進行中，海、空軍雙方要發射炸彈攻擊對方時，有一個限制即：攻擊之一方必須二種飛機或船隻均在螢幕上，才能成功的發射武器。且發射的順序爲：在海軍方面第一枚飛彈必定由驅逐艦發射，若第一枚飛彈未在螢幕消失前又再發射一枚飛彈，才由潛艇所發射；至於空軍第一枚炸彈由戰鬥機所投射，第二枚才由轟炸機所投射。即在螢幕上任何一方同時

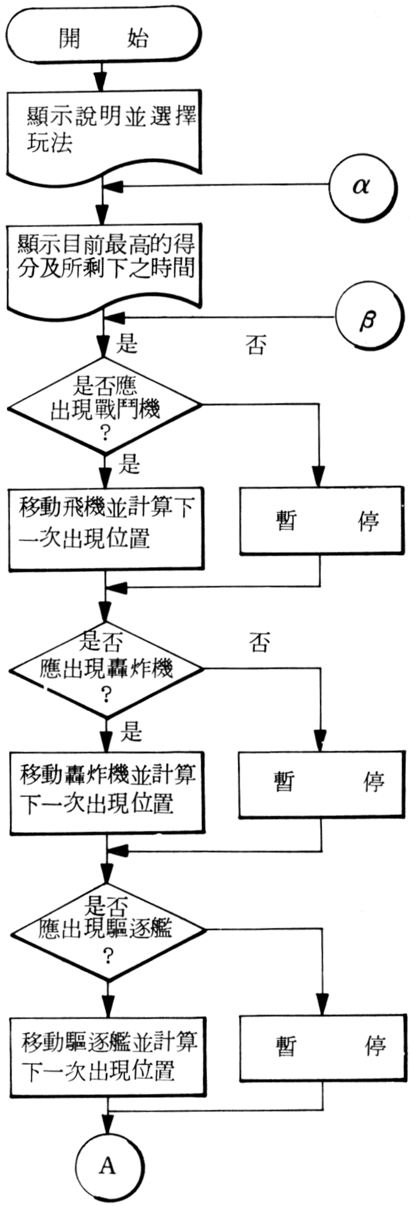
最多只能發射兩枚武器，若螢幕上各方同時僅發射出一枚投射武器，則必爲戰鬥機或驅逐艦所發出。且海、空軍雙方在螢幕上出現的第二枚飛彈或炸彈會改成直接由PADDLE控制其飛行方向，使得其成爲一種導向魚雷或導向飛彈，以增加遊戲的複雜度。

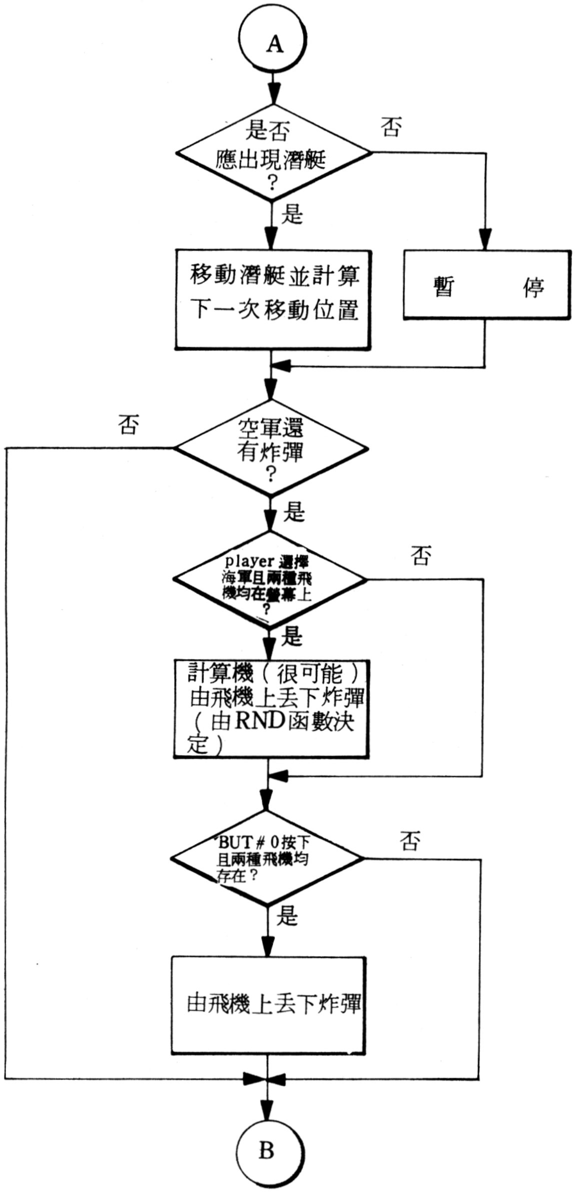
程式在畫面的處理方雖然是採用低解像圖面，但在一些畫面效果方面卻做得很仔細，如整個畫面畫分爲三部份：天空、淺海、深海，飛機在天空部份移動，驅逐艦則在海面上移動，而潛艇則在深海與淺海中移動。且炸彈碰到海面時引起水花四濺，船隻和飛機被擊中時的爆炸均有處理，再加上數種音響效果的配合，可算是一個處理的很仔細的程式。

程式流程

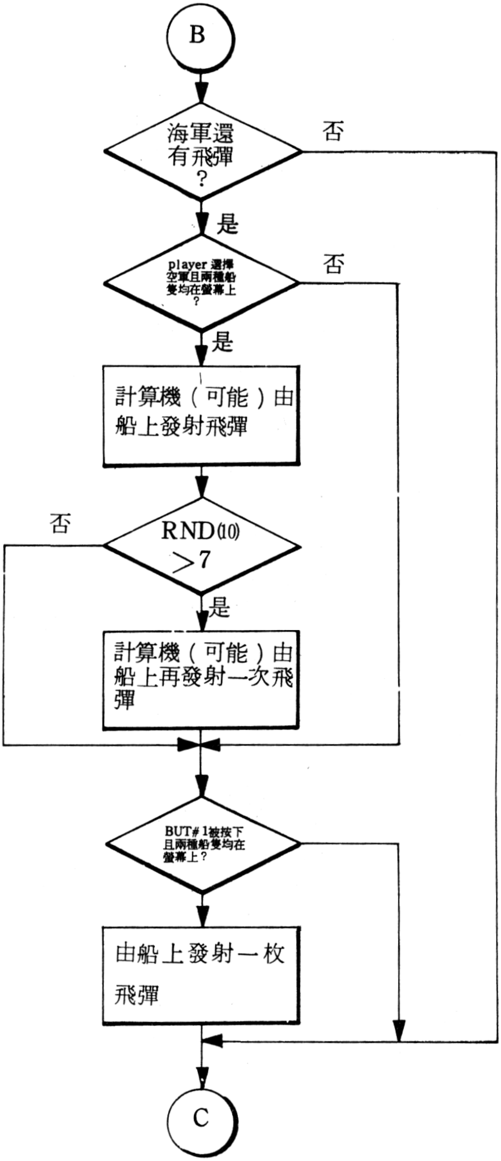
在本程式中，大部份行數均擁有數個敍述句（statements），如此做法，正如我們在第二章所說的可增加執行上的速度。又程式中將常常用到的一些繪圖或清除圖形的副程式，置於程式的開頭（第10，11，12，13，14 行），也是爲了增加執行上的速度。甚至程式第一行就GOSUB 6000 設定變數也可以說是爲了加快執行速度（但在本程式中影響不大），此外將炸彈／飛彈之描繪置於程式之前面部份（此程式中置於215，300，320，400，500，520）也能加快一些速度。

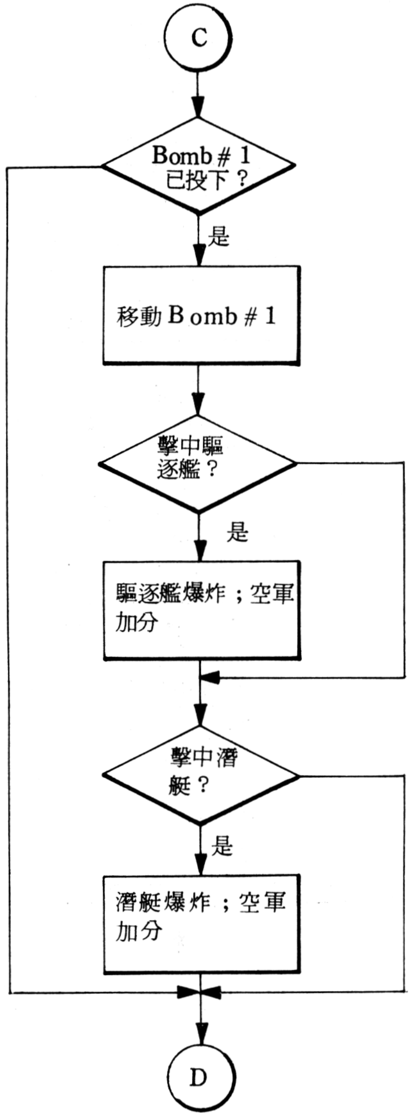
上面只是程式在整個安排上的一些問題，至於程式的流程，則以縮減的型式列於次頁：


開　始
顯示說明並選擇玩法
α
顯示目前最高的得分及所剩下之時間
β
是
否
是否應出現戰鬥機？
是
移動飛機並計算下一次出現位置
暫　停
否
是否應出現轟炸機？
是
移動轟炸機並計算下一次出現位置
暫　停
是否應出現驅逐艦？
移動驅逐艦並計算下一次出現位置
暫　停
A

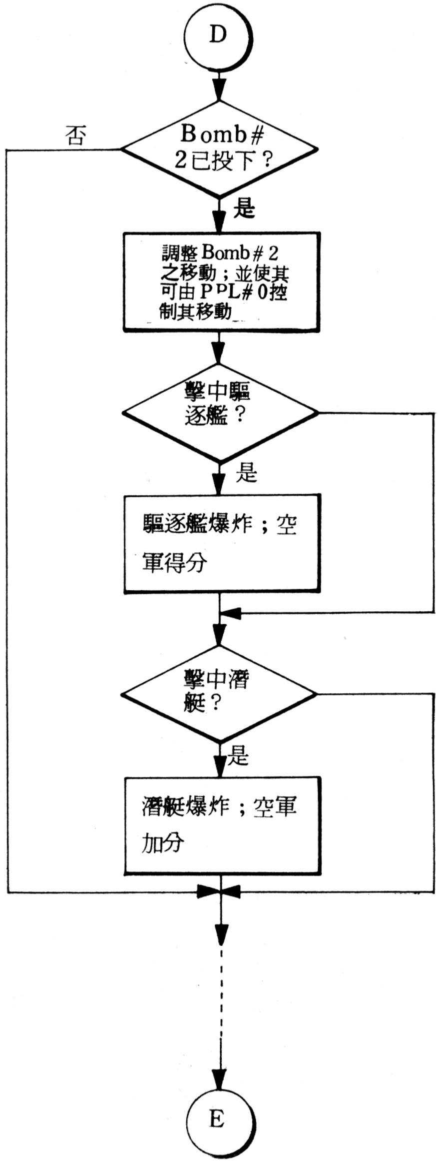

A
是否
應出現潛艇
?
否
是
移動潛艇並計算
下一次移動位置
暫　停
空軍還
有炸彈
?
否
是
player 選擇
海軍且兩種飛
機均在螢幕上
?
否
是
計算機（很可能）
由飛機上丟下炸彈
（由RND函數決
定）
BUT # 0按下
且兩種飛機均
存在?
否
是
由飛機上丟下炸彈
B

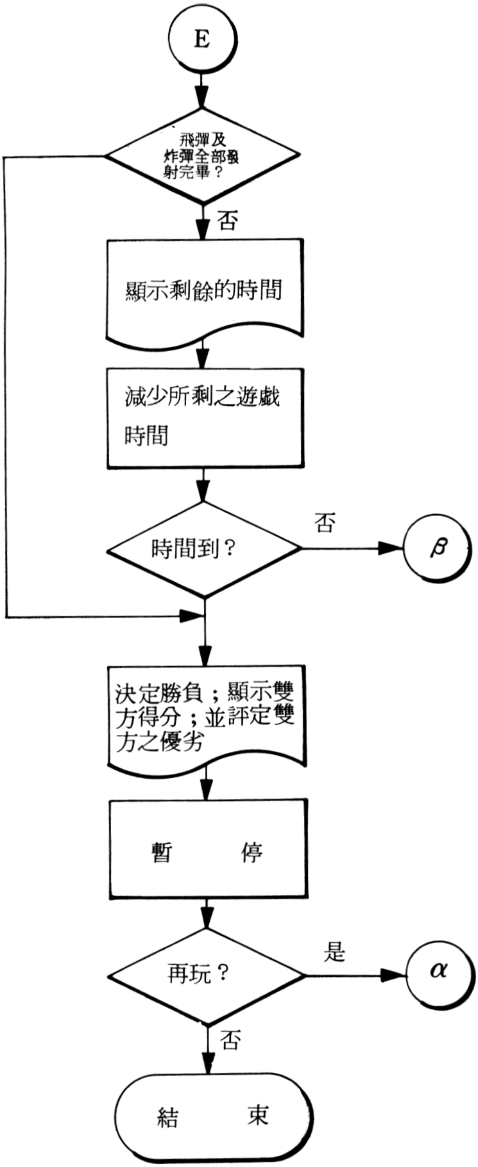
程式流程圖：（CONT）


B
海軍還
有飛彈
?
否
是
player 選擇
空軍且兩種船
隻均在螢幕上
?
否
是
計算機（可能）由
船上發射飛彈
否
RND(10)
>7
是
計算機（可能）由
船上再發射一次飛
彈
BUT# 1被按下
且兩種船隻均在
螢幕上？
由船上發射一枚
飛彈
C


C
Bomb # 1
已投下？
是
移動 B omb # 1
擊中驅
逐艦？
是
驅逐艦爆炸；空軍
加分
擊中潛
艇？
是
潛艇爆炸；空軍
加分
D

程式流程圖：（ CONT ）


D
Bomb#
2已投下？
否
是
調整Bomb#2
之移動；並使其
可由PPL#0控
制其移動
擊中驅
逐艦？
是
驅逐艦爆炸；空
軍得分
擊中潛
艇？
是
潛艇爆炸；空軍
加分
E


E
飛彈及炸彈全部發射完畢？
否
顯示剩餘的時間
減少所剩之遊戲時間
時間到？
否
β
決定勝負；顯示雙方得分；並評定雙方之優劣
暫　停
再玩？
是
α
否
結　束

程式中主要變數說明

程式中使用到許多變數，但大部份所代表的意義均很簡單：

(1) A$： 當遊戲者選擇與機器比賽時，此字串變數即用以儲存遊戲者爲海軍或空軍，A$=N（海軍NAVY），A（空軍AIRFORCE）。此變數同時可用以決定，比賽的進行是由APPLE與遊戲者比賽或爲二位遊戲者之間的比賽（若爲後者A$=“ ”）。

(2) PXN，PYN ：戰鬥機（FIGHTER）將出現的新位置座標；
PXO，PYO ：戰鬥機出現的原來位置座標；
BXN，BYN ：轟炸機（BOMBER）將出現的新位置座標；
BXO，BYO ：轟炸機出現的原來位置座標；
DXN，DYN ：驅逐艦（DESTROYER）將出現的新位置座標；
DXO，DYO ：驅逐艦（SUBMARINE）出現的原來位置座標；
SXN，SYN ：潛艇將出現的新位置座標；
SXO，SYO ：潛艇出現的原來位置座標；
BX1，BY1 ：炸彈1（BOMB#1）的位置座標；
BX2，BY2 ：炸彈2（BOMB#2）的位置座標；
MX1，MY1 ：飛彈1（HISSILE#1）的位置座標；
MX2，MY2 ：飛彈2（MISSILE#2）的位置座標；
ASC，NVY ：分別爲空軍（AIRFORCE）與海軍（NAVY）的得分。
BMS，SHS ：分別爲炸彈（BOMBS）與飛彈（MISSILES）還剩下的數目，其初值均爲15。
FXN，FYN ：在呼叫繪圖的副程式時，必須指出圖形的位置，此二變數即指示下一次繪圖的座標位置。

FXO，FYO：　指示前一個圖形的繪出位置座標。連同前二個變數，均爲繪圖副程式的參數，欲清除或繪出的圖形，其位置座標在呼叫繪圖副程式時必須先存入此類變數之中。

(3)CF，CB，CD，CS，EXPL，CL：以上分別指示戰鬥機，轟炸機、驅逐艦、潛水艇，及爆炸時火花之顏色又CL代表（（COLOR OF LIGHT BLUE）淺海的顏色（淡藍色）。這些變數均爲繪圖副程式在畫圖時所畫出的顏色，故其值分別表爲：68＝$44，34＝$22，170＝$AA，255＝$FF，17＝$11，119＝$77，以配合低解像圖形顏色之表示法。而另一些變數LT，DK，DJ，SJ，雖然代表的顏色也分別爲：淺藍色，深海的顏色、驅逐盤的顏色及潛艇的顏色，但並不是用來做爲呼叫繪圖副程式的引數，而是直接利用BASIC的一些指令（如 COLOR ＝）繪圖或判斷圖形顏色，故可直接用值7，2，10，15 分別表示。

(4)PS，SS：分別表示戰鬥機和潛水艇的移動速度，在每一次戰鬥機或潛艇剛出現時，會利用隨機函數RND（3）＋1求出每次出現時的移動速度。

(5)PCT，BCT，DCT，SCT：爲 FLIGHT、BOMBER、DESTROYER、SUBMARINE COUNT DOWN。這四個變數用來代表四種武器出現的時間間隔，每當上述四種武器的任何一種自螢幕消失後，便利用隨機函數RND（10）計算下一次出現的間隔時間。如 PCT＝6時，戰鬥機在螢幕上不出現，但每次檢查時會將 PCT＝PCT－1（COUNT DOWN）直到 PCT＝0，才在螢幕上顯示戰鬥機。利用這種方式，可使得上述四種武器以任意時間間隔出現，再加上PS，

SS 變數的使用，更使得戰鬥機及潛艇移動速度不定，讓遊戲者無法事先描準。

(6) ZS，ZH：爲炸彈2（BOMB＃2）與飛彈2（BOMB＃2）的水平移動變量。炸彈2及飛彈2與炸彈1及飛彈1有極大的不同：炸彈1及飛彈1飛行時僅改變垂直位置座標而不改變水平座標，故其運動軌跡爲垂直的向上或向下線段；但炸彈2及飛彈2除了根據炸彈（或飛彈）對目標物的相對位置決定水平飛行方向外（即ZS，ZH設定爲1或－1。當飛彈或炸彈在目標物之後方則設定爲1，以便追趕，當在目標物之前，則設定爲－1，以便能迎頭碰上。當然前後乃對目標物的行進方向而言），遊戲者並可藉著DADDLE＃0或＃1改變炸彈2及飛彈2的移動方向。

(7) COL，SHAPE：這兩個變數均爲呼叫低解像繪圖副程式的參數。COL爲所繪圖形的顏色，SHAPE 則爲代表圖形的圖形表所在位址。程式中一共有六個低解像圖形表即：戰鬥機、轟炸機、驅逐艦、潛艇、濺起的水花及爆炸的圖形，其起始位址分別爲＄0900，＄0910，＄0920，＄0930，＄0940，＄0950。而SHAPE中所存的即爲這些位址的低位元組值，如對轟炸機而言，SHAPE＝＄10＝16。

(8)低解像繪圖副程式所用到的一些位址：位址＄8 儲存圖形的顏色；位址＄5，＄6 爲圖形在螢幕出現的位置座標，＄5 爲X座標，＄6爲Y座標；＄803（＝2051）爲圖形表的低位元組位址，即SHAPE中之值。這些位址設定之後便可CALL 2048 呼叫繪圖副程式，執行繪圖工作。

(9)發聲副程式所用到的一些位址：＄981（2433）儲存聲音的長短（DURATION），＄98F（2447）儲存聲音之高低，發

聲程式本身則存於＄980至＄995的記憶區中，故將上述位址設定後，呼叫＄980（CALL 2432）便可發出指定的聲音。（上述副程式請見後面所附程式）。

程式流程說明（分爲副程式及主流程二部份介紹）

本程式除了 INTEGER BASIC 程式之外，還有二個機器語言副程式及圖形表。由於程式中假設遊戲者具有磁碟裝置。因此在程式中開頭：

0　POKE 204，0：POKE 205，16：POKE 74，0：POKE 75，16　設定　LOMEM：4096（INTEGER 沒有 LOMEM：見附錄一）

1　D＄＝“ ”：REM CTRLD：PRINT D＄；“BLOAD AIRSEA·B”

將各機器語言副程式由磁碟中讀出存入指定位址。

因此對於沒有磁碟機的遊戲者，則須以下列步驟進行：
⑴設定LOMEM：4096，⑵由磁帶中存入機器語言副程式；⑶ RUN。

㈠：以下將程式中所使用的副程式列出說明：

⑴ 7000-7075 行：爲顯示遊戲說明及選擇單人玩或雙人比賽的副程式。

⑵ 6000-6040 行：6000-6023 行繪出圖形背景，6025-6040 行設定程式變數（見前面的各變數說明）

⑶ 8000-8020 行：此副程式僅在單人遊戲時才執行，供遊戲者選擇加入海軍或空軍。

⑷ 10-14 行：其作用爲繪出指定圖形。主要包括四部份。

① 10-11 行：第 10 行用以除去指定位置上的圖形（利用黑色重畫一遍），第 11 行在新位置上按指定顏色畫出圖形。這

兩行程式主要是用以移動戰鬥機、轟炸機及驅逐艦之用（因其背景爲黑色）。

②第 12 行：此行程式作用爲在指定位置將圖形清除爲黑色。此行程式是當圖形移動至螢幕邊緣時，只將圖形清除而不繪出圖形。

③第 13 行與 11 行：第 13 行將圖形清除爲底色後，再至 11 行繪出新的圖形。但 13 行所使用的底色爲淡藍色 ，爲螢幕中淺海的顏色，故這二行程式僅供移動潛水艇之用。

④第 14 行： 主要作用爲將圖形清除爲底色，由於底色爲淡藍色，故僅供當潛艇到達螢幕邊界時消失之用。

⑤ 84-85 行：戰鬥機已移動至螢幕邊緣或爆炸時，在消失前設定 ：下次出現的時間間隔（PCT）、下次出現時的移動速度（PS），下次出現位置（PXN＝3），並將圖形清除。

(6) 89-90 行作用同(5)只是對像改爲轟炸機，且其移動速度固定，故不須使用隨機函數求下次出現時的移動速度。180-185 行作用同 89-90 行但對像爲驅逐艦。 188-190 行作用同(5)只是使用對像爲潛水艇。

(7) 900 行及 950 行：這兩個副程式，只有當機器與遊戲者比賽時（單人遊戲）才會被呼叫。 900行爲機器爲空軍，利用隨機函數 RND(5)決定當時是否要投下炸彈。950行爲機器爲海軍時，利用隨機函數RND(5)決定是否要發射飛彈攻擊空軍（遊戲者）。故在一個回合的比賽中這兩個副程式只利用到其中的一個。

(8) 999-1200 行：投下炸彈（空軍）。程式在 1000 及 1010 行分別測試炸彈 1 及炸彈 2 是否還在螢幕上（已投下），若已投下則不再投下新的炸彈，否則一次僅投下一枚炸彈。若投

下的為炸彈1則根據戰鬥機的位置決定炸彈1投下的位置（第1007行），若為炸彈2則根據轟炸機的位置決定炸彈2投下的位置（第1011 行）。故可知炸彈1均由戰鬥機所投下而炸彈2則均由轟炸機所投下。

(9) 1049-1070行：發射飛彈（海軍），其功能與(8)類似。且飛彈1由驅逐艦所發射，飛彈2則由潛艇所發射。此外還必須注意到，無論投彈或發射飛彈均有聲音配合且投下及發射兩種動作所發出的聲音不同。

(10) 1099-1110行：為處理驅逐艦被擊中爆炸的副程式。其中呼叫的5000行副程式用以製造爆炸的效果（見後）。

(11) 1199-1225行：為處理潛艇被擊中爆炸的副程式。

(12) 1299-1350行：為處理轟炸機被飛彈擊中爆炸的副程式。

(13) 1399-1430行：為處理戰鬥機被飛彈擊中爆炸的副程式。

(14) 1499-1545行：此副程式處理當飛彈飛到螢幕頂端時的爆炸效果。1500行利用白色（POKE 8，255）畫出白色爆炸圖形，再利用1517行利用黑色將爆炸圖形清除。1515行則用以發出爆炸聲音。此外1535行決定是飛彈1或飛彈2爆炸，將其Y座標設定為0，使1050 行的副程式可再發射飛彈。

(15) 1599-1650行：處理當炸彈落下碰到水面時水花四濺的效果，由於水花為淺海顏色故設定顏色為淡藍色（LT）。

(16) 1700行及1710行：分別處理炸彈1及炸彈2沈至深海時，結束程式對炸彈的控制（因深海顏色與炸彈顏色相同）。

(17) 4999-5050行：處理爆炸的效果。5000行顯示爆炸圖形，5020行發出爆炸聲音，5025及5030行清除爆炸圖形，但必須考慮若為潛艇爆炸時（FYN ＞ 23 ），由於爆炸地點在海中，故底色必須設定為淡藍色。此外5033 行檢查若為潛

艇爆炸則呼叫 5050 行副程式將淺海中潛艇出現範圍（X座標 25-29）用淡藍色重新畫一次，藉以消除潛艇的圖形。

(二)主程式流程：

主程式由第 8 行開始至 825 行結束，其中夾雜著數個副程式，因此程式結構看起來有點亂。事實上在本程式中對各副程式的安置均經過一番考慮，按照其被呼叫次數的多寡，分別安置在程式前面部份或後面，以求能獲得最快的執行速度。

關於程式的流程，已在前面介紹過了，以下對其中較難的部份說明：

(1) 25-45 行：首先檢查戰鬥機是否應出現或已出現在螢幕上（IF PCT＝0），是則移動飛機，並按其飛行速度（PS）決定下一次出現的位置，及測試是否已飛至螢幕邊緣。若PCT＞0代表還不到戰鬥機出現的時機，故僅將PCT減1，至PCT為0時飛機才出現。以下 50-75 行、95-125 行、及 145-175 行分別對轟炸機、驅逐艦及潛艇做同樣方式的處理。

(2)第 200 行利用 BMB，SHL記錄 PUSHBUTTON＃0與＃1是否被按下。若 A$＝″N″即代表為單人遊戲且遊戲者為海軍，此時若機器所屬空軍的兩種飛機均出現在螢幕上則至 900 行的副程式，利用RND 決定是否為空軍投彈的使機。205 行則為雙人遊戲或單人遊戲者為空軍時的處理。

(3) 207 及 208 行為單人遊戲且機器為海軍時決定是否發射飛彈的程式；210 行則處理雙人或單人遊戲時，遊戲者為海軍時有關飛彈的發射。在(2)、(3)兩部份我們注意到程式中的一個限制即：發射投射武器的一方，必須兩種飛機或船隻均出現在螢幕上時才能成功的投射武器。事實上這個限制只是為了程式在

撰寫上的方便，避免一些較複雜的情形，如發射第二枚飛彈時，僅有一艘船在螢幕上，則此飛彈必須由此船隻發射，但有時二艘船均在螢幕上，又要由沒有發射飛彈的船隻發射，如此會使 1000 行及 1050 行的副程式變得較複雜，減慢遊戲進行的速度。

(4) 214-245 行：爲對炸彈 1 的控制。215 行根據目前炸彈的位置，顯示炸彈的移動，235 行由 SCRN（BX1，BY1+1）=10，利用顏色判斷炸彈 1 是否已擊中驅逐艦（顏色碼 10 爲驅逐艦的顏色）。245 行若 BY1 = 22 代表炸彈 1 已撞至水面此時應呼叫 1600 行副程式，顯示水花四濺的效果，若 BY1 = 30 代表炸彈 1 已落至深海，呼叫 1700 行的副程式，結束對炸彈 1 的處理。

(5) 300-337 行：爲對炸彈 2 的控制。315 行根據炸彈與潛艇的相對位置決定炸彈 2 的移動方向，以能擊中潛艇的方向爲方向。317 行則再根據 PADDLE # 0 變動炸彈的移動方向。

(6) 399-426 行爲對飛彈 1 的控制，方式類似(4)所討論的內容。500-535 則爲對飛彈 2 的控制，方式亦類似(5)所討論的內容故不再贅述。

(7) 600 行檢查程式是否應提早結束，因雙方武器均已用盡（BMS = 0，SHS = 0）且攻擊亦已結束（MY1 =MY2 = BY1 = BY2=0），只好提早結束。

(8) 700-825 行：對空軍及海軍的成績做一番評估，並顯示分數判斷勝負。

以下爲程式中所使用到的低解像圖形表（程式部份見後）

1. 戰鬥機（FIGHTER）

0900-0A 0B 0C 0C 00 00

2. 轟炸機（BOMBER）

0910-0C 0C 0C 0C 0A 0B 00

3. 驅逐艦（DESTROYER）

0920-0C 0C 0A 0B 0C 0A 0A 0A

0928-0B 0B 0B 0C 00

4. 潛水艇（SUBMARINE）

0930-0C 0C 0C 0D 0A 00

5. 水花（SPLASH）

0940-0D 01 0D 01 0D 03 03 03

0948-0C 02 0D 00

6. 爆炸（EXPLOSION）

0950-0C 03 0C 01 0D 01 0C 04

0958-0D 00

```
>LIST
    0 POKE 204,0: POKE 205,16: POKE
      74,0: POKE 75,16
    1 D$="": PRINT D$;"BLOAD AIRSEA.B"

    2 REM   *************************
    3 REM   ** INTEGER BASIC ONLY  **
    4 REM   ** LOWE RESOLUTION GR. **
    5 REM   **                     **
    6 REM   ** COPYRIGHT 1979      **
    7 REM   *************************
    8 CALL -936: GOSUB 7000: CALL
      -936: GR : GOSUB 6000: IF Z$
      ="Y" THEN GOSUB 8000
    9 VTAB 24: IF PHS>0 THEN PRINT
      "PREV HI SCORE=";PHS;: TAB
      20: PRINT "TIME LEFT =";CTD;
      : VTAB 20: GOTO 25
   10 POKE 8,0: POKE 5,FX0: POKE
      6,FY0: POKE 2051,SHAPE: CALL
```

```
   2048: REM  EASE OLD SHAPE
11 POKE 8,COL: POKE 5,FXN: POKE
   6,FYN: POKE 2051,SHAPE: CALL
   2048: RETURN : REM   DRAW NEW SH
   APE
12 POKE 8,0: POKE 5,FXN: POKE
   6,FYN: POKE 2051,SHAPE: CALL
   2048: RETURN : REM  EASE-END OF
   LINE
13 POKE 8,CL: POKE 5,FXO: POKE
   6,FYO: POKE 2051,SHAPE: CALL
   2048: GOTO 11: REM  EASE OLD LIT
   E BLUE
14 POKE 8,CL: POKE 5,FXO: POKE
   6,FYO: POKE 2051,SHAPE: CALL
   2048: RETURN : REM  EASE OLD LIT
   E BLUE
15 REM  << REQUIRES INTEGER BASIC >
   >
23 REM  FLIGHTER PLANE ROUTINE
25 IF PCT>0 THEN 45:SHAPE=0:COL=
   CF:FXO=PXO:FYO=PYO:FXN=PXN:
   FYN=PYN: GOSUB 10:PXO=PXN:PXN=
   PXN+PS: IF PXN>=34 THEN GOSUB
   85
45 IF PCT>0 THEN PCT=PCT-1
48 REM  ** BOMBER ROUTINE
50 IF BCT>0 THEN 75:SHAPE=16:COL=
   CB:FXO=BXO:FYO=BYO:FXN=BXN:
   FYN=BYN: GOSUB 10:BXO=BXN:BXN=
   BXN-1: IF BXN<=2 THEN GOSUB
   90
75 IF BCT>0 THEN BCT=BCT-1
80 GOTO 95
84 REM   ** RESET FLIGHTER AND COUN
   TDOWN
85 PXN=3:PCT= RND (10): GOSUB
   12:PS= RND (3)+1:FXN=FXO:FYN=
   FYO: GOSUB 12: RETURN
89 REM  ** RESET BOMBER AND COUNTDO
   WN
90 BXN=34:BCT= RND (10): GOSUB
   12:FXN=FXO:FYN=FYO: GOSUB 12
   : RETURN
94 REM  ** DESTROYER ROUTINE
```

```
 95 IF DCT>0 THEN 125:SHAPE=32:
    COL=CD:FXO=DXO:FYO=DYO:FXN=
    DXN:FYN=DYN: GOSUB 10:DXO=DXN:
    DXN=DXN+1: IF DXN>=34 THEN
    GOSUB 185
125 IF DCT>0 THEN DCT=DCT-1
144 REM  SUBMARINE ROUTINE
145 IF SCT>0 THEN 175:SHAPE=48:
    COL=CS:FXO=SXO:FYO=SYO:FXN=
    SXN:FYN=SYN: GOSUB 13
146 SXO=SXN:SXN=SXN-SS: IF SXN<=
    0 THEN GOSUB 190
175 IF SCT>0 THEN SCT=SCT-1: GOTO
    200
180 REM  ** RESET DESTROYER AND COUN
    TDOWN
185 DXN=1:DCT= RND (10): GOSUB
    12: RETURN
188 REM   **RESET SUB AND COUNTDOWN
190 FXO=SXO: GOSUB 14:SCT= RND
    (10):SXN=35:SS= RND (3)+1: RETURN
199 REM  ** FIRING SEQUENCE
200 BMB= PEEK (-16287):SHL= PEEK
    (-16286): IF BMS=0 THEN 207
    : IF A$="N" AND PXN>4 AND PXN<
    35 AND BXN>4 AND BXN<34 THEN
    GOSUB 900
205 IF BMB>127 AND PXN>5 AND BXN<
    34 THEN GOSUB 1000
207 IF SHS=0 THEN 215: IF A$="A"
     AND DXN>7 AND DXN<35 AND SXN>
    7 AND SXN<35 THEN GOSUB 950

208 IF A$="A" AND RND (10)>7 AND
    DXN>7 AND DXN<35 AND SXN>7 AND
    SXN<35 THEN GOSUB 950
210 IF SHL>127 AND DXN>3 AND SXN<
    34 THEN GOSUB 1050
214 REM  ** BOMB NUMBER 1
215 IF BY1=0 THEN 300: COLOR=0:
     IF BY1>=23 THEN COLOR=LT: PLOT
    BX1,BY1:BY1=BY1+1: COLOR=5:
     PLOT BX1,BY1
235 IF SCRN(BX1,BY1+1)=10 THEN
    GOSUB 1100: IF SCRN(BX1,BY1+
```

```
    1)=15 THEN GOSUB 1200:FXN=BX1:
    FYN=BY1
240 IF SCRN(BX1,BY1+1)=15 THEN
    GOSUB 1200
245 IF BY1=22 THEN GOSUB 1600: IF
    BY1=30 THEN GOSUB 1700
299 REM  ** BOMB NUMBER 2
300 IF BY2=0 THEN 400: COLOR=0:
     IF BY2>=23 THEN COLOR=LT: PLOT
    BX2,BY2:BY2=BY2+1
315 IF A$="N" AND BX2<SXN+2 THEN
    ZS=1: IF A$="N" AND BX2>SXN+
    2 THEN ZS=-1:BX2=BX2+ZS
316 IF A$="N" THEN 320
317 IF PDL (0)>160 THEN BX2=BX2+
    1: IF PDL (0)<80 THEN BX2=BX2-
    1
320 IF BX2<4 THEN BX2=4: IF BX2>
    36 THEN BX2=36: COLOR=5: PLOT
    BX2,BY2
325 IF SCRN(BX2,BY2+1)=DJ THEN
    GOSUB 1100: IF SCRN(BX2,BY2+
    1)=SJ THEN GOSUB 1200
332 FXN=BX2:FYN=BY2
335 IF BY2=22 THEN GOSUB 1600
337 IF BY2=30 THEN GOSUB 1710
399 REM  ** TEST MISSILE #1 HIT**
400 IF MY1=0 THEN 500: COLOR=0:
     IF MY1>=23 THEN COLOR=LT: PLOT
    MX1,MY1:MY1=MY1-1: COLOR=1:
     PLOT MX1,MY1
425 IF SCRN(MX1,MY1-1)=2 THEN GOSUB
    1300: IF SCRN(MX1,MY1-1)=4 THEN
    GOSUB 1400:FXN=MX1:FYN=MY1:

426 IF MY1<=4 THEN GOSUB 1500
499 REM  ** TEST MISSLE #2 HIT **
500 IF MY2=0 THEN 600: COLOR=0:
     IF MY2>=23 THEN COLOR=LT: PLOT
    MX2,MY2:MY2=MY2-1
515 IF A$="A" AND MY2<PXN THEN
    ZH=1: IF A$="A" AND MX2>PXN THEN
    SH=-1:MX2=MX2+SH
516 IF A$="A" THEN 519
517 IF PDL (1)>160 THEN MX2=MX2+
```

```
    1: IF PDL (1)<80 THEN MX2=MX2-
    1
519 IF MX2<1 THEN MX2=1: IF MX2>
    35 THEN MX2=35
520 COLOR=1: PLOT MX2,MY2: IF SCRN(
    MX2,MY2-1)=2 THEN GOSUB 1300
    : IF SCRN(MX2,MY2-1)=4 THEN
    GOSUB 1400
535 FXN=MX2:FYN=MY2: IF MY2<=4 THEN
    GOSUB 1500
599 REM  ** TEST END OF GAME **
600 IF BMS=0 AND SHS=0 AND BY1=
    0 AND BY2=0 AND MY1=0 AND MY2=
    0 THEN 700
602 VTAB 24: TAB 23: CALL -868:
     PRINT CTD;: VTAB 20:CTD=CTD-
    1: IF CTD=0 THEN 700
605 GOTO 25
699 REM  ** EVALUATE SCORE
700 CALL -936: VTAB 23: IF ASC>
    NVY THEN PRINT "** THE AIR FORCE
     WINS! **": IF NVY>ASC THEN
    PRINT "** THE NAVY WINS! **"

702 IF NVY>PHS AND NVY>ASC THEN
    PHS=NVY: IF ASC>NVY AND ASC>
    PHS THEN PHS=ASC
705 IF NVY=ASC THEN PRINT "** A TIE
    BATTLE! **": VTAB 21: PRINT
    "NAVY SCORE ";NVY;" RATES "
    ;:SCR=NVY
730 IF SCR<=3 THEN PRINT "AUNNER 4TH
     CLASS"
735 IF SCR>3 AND SCR<=6 THEN PRINT
    "GUNNER 2ND CLASS"
740 IF SCR>6 AND SCR<=10 THEN PRINT
    "GUNNER 1ST CLASS"
745 IF SCR>10 AND SCR<=14 THEN
    PRINT "CHIEF GUNNER"
750 IF SCR>14 AND SCR<=17 THEN
    PRINT "EXEC OFFICER"
755 IF SCR>17 AND SCR<=20 THEN
    PRINT "CAPTAIN!"
760 IF SCR>20 THEN PRINT "ADMIRAL!"
```

```
770 VTAB 22: PRINT "AIR FORCE SCORE
    ";ASC;" RATES ";
775 IF ASC<=3 THEN PRINT "BOMBER 4TH
     CLASS"
780 IF ASC>3 AND ASC<=6 THEN PRINT
    "BOMBER 2ND CLASS"
785 IF ASC>6 AND ASC<=10 THEN PRINT
    "BOMBER 1ST CLASS"
790 IF ASC>10 AND ASC<=14 THEN
    PRINT "CHIEF BOMBER"
795 IF ASC>14 AND ASC<=17 THEN
    PRINT "EXEC OFFICER"
800 IF ASC>17 AND ASC<=20 THEN
    PRINT "WING CAPTAIN!"
805 IF ASC>20 THEN PRINT "GENERAL!"

807 FOR Y=1 TO 1000: NEXT Y:ZS=
    0:ZH=0
810 VTAB 24: INPUT "HIT RETURN TO FL
    Y/SAIL ANOTHER MISSION",A$:
     GOTO 2
825 END
900 NSHOOT= RND (5): IF NSHOOT<=
    1 THEN GOSUB 1000: RETURN
950 ASHOOT= RND (5): IF ASHOOT<=
    1 THEN GOSUB 1050: RETURN
999 REM   ** START NEW BOMB DROP
1000 IF BY1>0 THEN 1010:BMS=BMS-
    1: POKE 2433,1: POKE 2447,200
    : CALL 2432
1007 BX1=PXO+1:BY1=PYO: VTAB 21:
     TAB 20: CALL -868: PRINT "BOMBS
     LEFT=";BMS: RETURN
1010 IF BY2>0 THEN 1020:BMS=BMS-
    1: POKE 2433,1: POKE 2447,200
    : CALL 2432
1011 BX2=BXO+1:BY2=BYO
1017 VTAB 21: TAB 20: CALL -868:
     PRINT "BOMBS LEFT=";BMS
1020 RETURN
1049 REM   ** START NEW MISSILE SHOT
1050 IF MY1>0 THEN 1060:SHS=SHS-
    1
1051 POKE 2433,1: POKE 2447,190:
     CALL 2432
1052 MX1=DXO:MY1=DYO
```

```
1057 VTAB 21: TAB 1: CALL -868: PRINT
     "MISSILE LEFT=";SHS;: TAB 20
     : PRINT "BOMBS LEFT=";BMS: RETURN
1060 IF MY2>0 THEN 1070:SHS=SHS-
     1: POKE 2433,1: POKE 2447,190
     : CALL 2432
1061 MX2=SXO:MY2=SYO
1067 VTAB 21: TAB 1: CALL -868: PRINT
     "MISSILE LEFT=";SHS;: TAB 20
     : PRINT "BOMBS LEFT=";BMS
1070 RETURN
1099 REM  ** SINK DESTROYER
1100 FXO=DXO:FYO=DYO:FXN=DXN:FYN=
     DYN: GOSUB 5000:ASC=ASC+1:SHAPE=
     32:FXN=DXO:FYO=DYN
1105 GOSUB 185: VTAB 23: TAB 20:
      PRINT "DESTROYER SUNK!": VTAB
     22: TAB 20: PRINT "AIRFORCE SCOR
     E=";ASC
1110 RETURN
1199 REM  ** SINK SUB
1200 FXO=SXO:FYO=SYO:FXN=SXN:FYN=
     SYN: GOSUB 5000:ASC=ASC+2:SHAPE=
     48: GOSUB 190
1205 VTAB 23: TAB 20: PRINT "SUBMARIN
     E SUNK!": VTAB 22: TAB 20: PRINT
     "AIRFORCE SCORE=";ASC
1225 RETURN
1299 REM  ** BOMBER SHOT DOWN
1300 FXO=BXO:FYO=BYO:FXN=BXN:FYN=
     BYN: GOSUB 5000:NVY=NVY+1:SHAPE=
     16: GOSUB 90
1305 VTAB 23: TAB 1: PRINT "BOMBER DO
     WN!": VTAB 22: TAB 1: PRINT
     "NAVY SCORE=";NVY: CALL 2048

1335 IF BY1>=30 THEN BY1=0: IF BY2>=
     30 THEN BY2=0
1350 RETURN
1399 REM  ** FIGHTER SHOT DOWN
1400 FXO=PXO:FYO=PYO:FXN=PXN:FYN=
     PYN: GOSUB 5000:NVY=NVY+2:SHAPE=
     0: GOSUB 85
1420 VTAB 23: TAB 1: PRINT "FIGHTER D
     OWN!": VTAB 22: TAB 1: PRINT
     "NAVY SCORE=";NVY
```

```
1430 RETURN
1499 REM  **FLAK ROUTINE
1500 POKE 5,FXN: POKE 6,FYN: POKE
     2051,80: POKE 8,255: CALL 2048

1515 POKE 2433,1: POKE 2447,150:
      CALL 2432
1517 COLOR=0: PLOT FXN,FYN: POKE
     2051,80: POKE 8,0: POKE 5,FXN:
      POKE 6,FYN: CALL 2048
1535 IF MY1<=4 THEN MY1=0: IF MY2<=
     4 THEN MY2=0
1545 RETURN
1599 REM  ** SPLASH ROUTINE
1600 POKE 5,FXN: POKE 6,FYN:SHAPE=
     64:COL=LT: GOSUB 11: GOSUB
     12
1616 POKE 2433,1: POKE 2447,150:
      CALL 2432
1650 RETURN
1700 COLOR=DK: PLOT BX1,BY1:BY1=
     0: RETURN
1710 COLOR=DK: PLOT BX2,BY2:BY2=
     0: RETURN
2100 IF SHL>127 AND DXN>4 AND SXN<
     33 THEN GOSUB 1050
4999 REM  ** EXPLOSION
5000 SHAPE=80: POKE 5,FXN: POKE
     6,FYN: POKE 8,EXPL: POKE 2051
     ,SHAPE: CALL 2048
5019 REM  ** EXPLOSION SOUND
5020 POKE 2433,4: POKE 2447,250:
      CALL 2432:
5024 REM  ** ERASE EXPLOSION
5025 POKE 2051,SHAPE: IF FYN<23 THEN
     POKE 8,0: IF FYN>23 THEN POKE
     8,CL
5030 POKE 5,FXN: POKE 6,FYN: POKE
     2051,SHAPE: CALL 2048
5033 IF BY1>23 OR BY2>23 THEN GOSUB
     5050
5035 RETURN
5050 COLOR=7: FOR N=25 TO 29: HLIN
     1,38 AT N: NEXT N: RETURN
6000 REM  ** DRAW OCEAN AND SET VAR
```

```
6005 COLOR=7: FOR Y=23 TO 29: HLIN
     0,39 AT Y
6010 NEXT Y
6015 COLOR=2: FOR Y=30 TO 39: HLIN
     0,39 AT Y: NEXT Y
6020 FOR Y=30 TO 39: HLIN 0,39 AT
     Y: NEXT Y
6022 COLOR=8:CTD=275
6023 HLIN 0,39 AT 0: VLIN 0,39 AT
     0: VLIN 0,39 AT 39: HLIN 0,
     39 AT 39
6025 BY1=0:BY2=0:MY1=0:MY2=0:BXO=
     0:BYO=5:BXN=33:BYN=5:FXO=3:
     FYO=3:FXN=4:FYN=3
6027 PXO=3:PYO=3:PXN=3:PYN=3
6030 DXO=0:DYO=22:DXN=1:DYN=22:SXO=
     35:SYO=29:SXN=34:SYN=29
6035 CF=68:CB=34:CD=170:CS=255:CL=
     119:CX=58
6038 LT=7:DK=2:DJ=10:SJ=15:EXPL=
     17
6040 PS=1:SS=1:PCT= RND (5):BCT=
      RND (5):DCT= RND (5):SCT= RND
     (5):ASC=0:NVY=0:BMS=15:SHS=
     15: RETURN
7000 CALL -936: REM  **INTRIDUCTION
7005 TAB 5: PRINT "THIS IS A AIR/SEA
     BATTLE!!"
7010 TAB 5: PRINT "AIR FORCE VS. THE
     NAVY"
7015 TAB 7: PRINT "A BOMBER IS WORTH
     1 HIT"
7020 TAB 7: PRINT "DESTROYER IS WORTH
      1 HIT"
7030 TAB 7: PRINT "SUBS AND FIGHTERS
      ARE WORTH 2"
7031 PRINT : PRINT "FIRING TWO MISSIL
     ES OR DROPPING"
7032 PRINT "TWO BOMBS AT THE SAME TIM
     E MAKES"
7033 PRINT "THE SECOND ONE A 'GUIDE'
     MISSILE"
7034 PRINT "AND PUTS IT UNDER PADDLE
     CONTROLL"
7035 PRINT : PRINT "EACH SIDE WILL BE
      JUDGE AT THE END"
```

```
7040 PRINT "OF THE MISSION"
7045 PRINT : PRINT "GOOD LUCK AND GOO
     D SHOOTING!!"
7050 PRINT : PRINT "IF YOU WANT TO PL
     AY AGAIN THE"
7055 PRINT " COMPUTER THEN TYPE 'Y',"

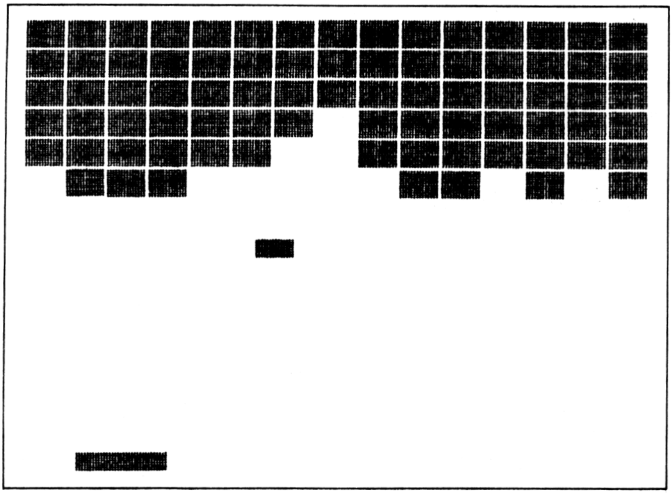
7060 PRINT "OTHERWISE TYPE OTHER LETT
     ER AND"
7065 INPUT "HIT RETURN",Z$
7075 RETURN
8000 CALL -936: PRINT "WHICH SERVICE
     WILL YOU JOIN ?"
8005 PRINT "ENTER 'A' FOR AIR FORCE O
     R": INPUT "ENTER 'N' FOR NAVY"
     ,A$: CALL -936
8020 RETURN
```

```
0980-   A0 04      LDY   #$04
0982-   A2 00      LDX   #$00
0984-   8A         TXA
0985-   18         CLC
0986-   E9 01      SBC   #$01
0988-   D0 FC      BNE   $0986
098A-   8D 30 C0   STA   $C030
098D-   E8         INX
098E-   E0 FA      CPX   #$FA
0990-   D0 F2      BNE   $0984
0992-   88         DEY
0993-   D0 ED      BNE   $0982
0995-   60         RTS
```

```
0800-   A2 00      LDX   #$00
0802-   BD 00 09   LDA   $0900,X
0805-   E8         INX
0806-   C9 01      CMP   #$01
0808-   F0 24      BEQ   $082E
```

```
080A-   C9 02      CMP   #$02
080C-   F0 25      BEQ   $0833
080E-   C9 03      CMP   #$03
0810-   F0 26      BEQ   $0838
0812-   C9 04      CMP   #$04
0814-   F0 27      BEQ   $083D
0816-   C9 0A      CMP   #$0A
0818-   F0 28      BEQ   $0842
081A-   C9 0B      CMP   #$0B
081C-   F0 29      BEQ   $0847
081E-   C9 0C      CMP   #$0C
0820-   F0 2A      BEQ   $084C
0822-   C9 0D      CMP   #$0D
0824-   F0 2B      BEQ   $0851
0826-   C9 00      CMP   #$00
0828-   F0 03      BEQ   $082D
082A-   4C 02 08   JMP   $0802
082D-   60         RTS
082E-   C6 06      DEC   $06
0830-   4C 56 08   JMP   $0856
0833-   E6 06      INC   $06
0835-   4C 56 08   JMP   $0856
0838-   E6 05      INC   $05
083A-   4C 56 08   JMP   $0856
083D-   C6 05      DEC   $05
083F-   4C 56 08   JMP   $0856
0842-   C6 06      DEC   $06
0844-   4C 64 08   JMP   $0864
0847-   E6 06      INC   $06
0849-   4C 64 08   JMP   $0864
084C-   E6 05      INC   $05
084E-   4C 64 08   JMP   $0864
0851-   C6 05      DEC   $05
0853-   4C 64 08   JMP   $0864
0856-   A9 00      LDA   #$00
0858-   85 30      STA   $30
085A-   A4 05      LDY   $05
085C-   A5 06      LDA   $06
085E-   20 00 F8   JSR   $F800
0861-   4C 02 08   JMP   $0802
0864-   A5 08      LDA   $08
0866-   85 30      STA   $30
0868-   A4 05      LDY   $05
086A-   A5 06      LDA   $06
086C-   20 00 F8   JSR   $F800
086F-   4C 02 08   JMP   $0802
```

程式七 打磚塊 (LITTLE BRICK OUT)



這是一個 APPLE Ⅱ 上很吸引人的遊戲，需要手與眼配合好才能有很高的成績。在螢幕上顯示有八行交錯排列的磚塊，每行的顏色不同得分也不同，而在螢幕左邊邊緣有一塊打板，用來打擊。主要玩法是每次玩時，有五個球，一開始發球由 APPLE 利用RND函數控制，任意將球由右往左投出，而遊戲者可以用PADDLE來控制打板往上或往下移動，來擋住球使它反彈，而球打到打板不同地方其反彈方向也就不同，若打板沒擋到這個球就出界。一次遊戲有五個球，表示五次沒擋到球就結束比賽。被球

打到的磚塊，就從螢幕上擦掉，因此磚塊剩下的越少分數也就越高，當然當磚塊被打光時，比賽也就結束，得分最高。在這個遊戲要拿高分也不是件難事，主要是要能控制好打板，注意球打到磚塊或上下二邊邊界及右邊邊緣反彈時的方向，小心擋球使球不要出界。相信只要多練習幾次，用心去玩每個人都能有很高的成績，並能體會到APPLE Ⅱ 的功能與打磚塊遊戲的樂趣。

本程式相信大家都玩過，但了解程式是如何寫出來的人相信不會太多。本文主要便是要與讀者共同研究本程式的寫法。了解本程式後對同類型的程式便能舉一反三了。

程式中主要變數說明

(1)A：爲顏色常數其値爲 2 ，表示深藍色（DARK BLUE）。

(2)B：爲顏色常數其値爲12，表示淡綠色（LIGHT GREEN）。

(3)C：爲顏色常數其値爲 9 ，表示橙色（ORANGE）。

(4)D：爲顏色常數其値爲13，表示黃色（YELLOW）。

(5)E：爲顏色常數其値爲 2 ，表示深藍色（DARK BLUE）。

(6)F：爲顏色常數其値爲15，表示白色（WHITE）。

(7)OQ：存放目前PADDLE（0）的輸出，其值範圍爲 0 到255。

(8)P：存放目前打板所在的位置，其值範圍爲 0 到34。由於螢幕的顯示是低解析度（LO—RES），打板是由 6 塊低解度點塊組成，並且畫面的顯示使用40×40，因此P最大值爲34

(9)Q：由APPLE（0）輸入的值，算出打板的位置時，所使用到的變數。Q的值首先是存入新的值，並與OQ比較求得新的座標範圍（0至34）然後再與P比較繪出新的打板並清除原有的打板。

(10)LL：表示打板的長度，其值爲5（實際長度6減1）。

(11) S ：表示目前的得分（SCORE　）。

(12) L ：表示目前還剩幾個球，一回遊戲有五個球。

(13) X ：表示球目前所在的X座標，範圍0至39。

(14) Y ：表示球目前所在的Y座標，範圍0至119。

(15) I ：表示球跳動後新的X座標，範圍0至39。

(16) J ：表示球跳動後新的Y座標，範圍0至119。

(17) K ：其值爲（J ／3），因螢幕顯示Y的範圍爲0至39故需將J除以3以便印出。

(18) W ：爲球跳動時，沿Y方向變動的大小。W初值是由RND隨機函數所產生，當碰到打板後不同位置其值也不同。W的值有（－3，－2，－1，1，3，5）六種。

(19) V ：爲球跳動時，沿X方向變動的大小，V初值爲－1，當碰到打板反彈時變成2，V的值有（-2，－1，2）三種。

(20) 位址789，790此兩個位址用來存放，已經玩過中得分最高的分數，其初值爲0。

(21) 位址787，388此兩個位址用來記錄，這次比賽是否爲第一次，當PEEK（787）＝37，PEEK（788）＝38時表示不是第一次比賽。

(22) A＄：存放遊戲者的英文名字。

(23) H＄：接受讀入回答"YES"或"NO"。

(24) VTB：其值由PEEK（37）得來，相當於VTAB的值，指示目前螢幕顯示在第幾行。

(25) TB：其值由PEEK（36）得來，相當於（CURSOR）所在的位置。

VTB與TB是用來控制，在螢幕上特定行特定列顯示文字，藉此控制畫面。

(26)ERASE: 當遊戲開始時，若遊戲者用錯PADDLE，用到 PADDLE1時，ERASE 的值將填入20，藉以控制在螢幕上顯示<<<PLEASE USE THE OTHER PADDLE >>>，提醒使用者使用正確的控制鈕。

(27)P 1 ：其值存放PDL(1)的值，範圍由0至255。

(28)OOLDP1：其值也是存放PDL(1)的值，但是用來與P 1比較，是否使用者用錯PADDLE。

程式中除了使用到上述各常數及變數之外，也使用到許多記憶體中的特定位址或直接呼叫監督程式中的幾個副程式。以下便將其列出並加以說明：

程式中使用到特殊APPLE位址與指令

(1)PDL(0)：讀取PADDLE0的值，範圍爲0至255。

(2)ABS(Q)：絕對值函數，其值爲 |Q|。

(3)SGN(Q)：正負號函數，當Q大於等於零其值爲1，而當Q小於零則爲0。

(4)SCRN(I,J)：取螢幕上座標（I，J）的顏色值。

(5)POKE 6,150：控制SPEAKER發聲的高低，範圍爲0至255。

POKE 7,3：控制SPEAKER發聲的長短，範圍爲0至255。

(6)PEEK(-16336)：位址－16336 爲SPEAKER，PEEK使它發聲。

(7)INT(K)：整數函數，其值爲K的整數部份。

(8)RND(1)：隨機函數，（RAMDOM NUMBER），其值在0與1之間。

(9)CALL-868：將CURSOR以後至右邊邊界清成空白。

(8)POKE 50,63：使螢幕顯示變成反白字（INVERSE）。

POKE 50,255 ：使螢幕顯示恢復正常字形（NORMAL）。

(9) POKE 2039,4 ：在螢幕顯示最後一行最後一字，存入"D"的反白字形，使螢幕畫面不會前進，保留畫面。

(10) PEEK（-16384)：讀取鍵盤上，所按入的字鍵。

(11) PEEK （-16287)：讀取按鈕（BUTTON 0）0是否被按下。

PEEK （-16286)：讀取按鈕（BUTTON 1）1是否被按下，其值大於127時表示已按下按鈕。

(12) CALL-936執行Home，清除螢幕，使CURSOR跳至左上角。

(13) CALL -958：將CURSOR以後至螢幕最後一行的最後一個字，清成空白。

(14) POKE-16368,0：表示目前所鍵入的字元，已經讀取過了可以接受下一個輸入。

(15) CALL 768：打磚塊程式，本身有一段組合語言副程式，用來發聲，其起始位址爲768，因此CALL 768，就是呼叫發聲副程式。

(16) 141 （$8D）：表示ASCII CODE "RETURN"。

155 （$9B)：表示ASCII CODE "ESC"。

160 （$A0)：表示ASCII CODE "SPACE"。

(17) PEEK(40)+PEEK(41)*256 ：40,41（BASL，BASH）爲監督程式所使用的位址，由上式可求出游標所在列的最左邊字元所對應的記憶位址。

(18) PEEK(37)：位址37存放VTAB的值。

PEEK(36)：位址36存放CURSOR所在的位置。

(19) LEN(A$)：檢查字串A$中共存放幾個字元，其值爲字串A$的字元數目。

(20) CHR$(A)：根據A值的大小，轉換成字ASCII CODE中的字

元，即該字元在ASCII CODE中排列的次序等於A值。故程式中的CHR$(34)即為〞〃〞雙引號。

(21)ASC(C)：傳回字元C所對應的ASCII碼。

(22)MID$(A$,I,J)：從字串A$中取出一字串出來。從A$第I個字元起取J個字元出來形成一長度為J的字串。

(23)LEFT(A$,I)：將字串A$從第一個字元起取I個字元，形成一長度I的字串。

程式流程說明

整個打磚塊程式，並不是容易的程式，為了說明方便起見，將程式分成17部份。這17部份有的較容易，有的較難，為了方便讀者解讀程式，本文對其中較難的部份加上額外說明，並以流程圖表示。

(一)程式架構：

(1)移動打板(MOVE PADDLE)20—70行，

(2)打磚塊(BRICK OUT)75—350行，

(3)繪出八行交錯排列磚塊(DRAW FIELD)355—450行，

(4)印出你贏了(CONGRATULATIONS YOU WIN)760—790行，

(5)印出分數評語(YOUR SCORE)800—1090行，

(6)按鈕後繼續比賽(PUSH BUTTON FOR NEW GAME)1100—1190行，

(7)印出每行磚塊被打掉的得分(PRINT POINT SCALE)1300—1390行，

(8)紅利部份(BONUS)1500—1690行，

(9)程式起始部份(START OF PROGRAM)2000—2335行

，

⑽印出說明（PRINT TITLE）2340 − 2485 行，

⑾設定初值（INITIALIZATIONS）2490 − 3020 行，

⑿按鈕後開始比賽（PUSH BUTTON TO BEGIN）4800行，

⒀印出請用另一個控制鈕（USE THE OTHER PADDLE）4810 − 4830 行，

⒁發聲副程式（SOUND ROUTINE）7000 − 7030 行，

⒂控制鈕沒有插上（PADDLE NOT PLUGGED IN）8500 行− 8600 行，

⒃按下結束鍵（″ESC″ KEY HAS BEEN PRESSED）11000 − 11110 行，

⒄結束（END）31111 行。

㈣程式說明

⑴移動打板（MOVE PADDLE）20−70行

此段副程式主要是依據PDL(0)的值，來移動打板的位置，也就是說旋轉PADDLE0 將使打板的位置跟著變動。PDL的值為0至255而打板的位置只有0至34，故將PDL的值除6所得的值取上限34下限0。上限取34是因為螢幕是以低解析度顯示，並且打板的長度為6，而縱座標只能使用0至39，（39− 6 + 1 =34），故上限為34。此段程式結束時，先在新的位置顯示打板，然後再將舊的打板清除。

⑵打磚塊（PLAY GAME）75 − 350 行

此段副程式是執行打磚塊，其主要結構為一迴圈，當磚塊統統打光了或者球沒擋到時便結束此迴圈。其主要執行過程如下：

(i)繪出新的球，並擦掉舊的球。

(ii)改換球的Y座標，即加上Y的變化量W，檢查新的Y座標是

否小於 0 或大於 119。大於 119 表示碰到底端，小於 0 表示碰到頂端。若發生這種情況時，將 W 變號使球方向改變，形成反彈。

(iii)改換球的 X 座標，即加上 X 的變化量 V。檢查新的 X 座標，若 X 小於 0 時表示球沒擋到，將跳出迴圈。

(iv)移動打板到新的位置。

(v)檢查 X 座標是否大於39，若是則表示球碰到右邊邊界，將 V 變號使球方向改變，形成反彈。

(vi)檢查螢幕球新的位置是否爲底色藍色，若是表示沒有東西，跳至步驟(i)形成迴圈。

(vii)若新的 X 座標等於 0，並且該位置之螢幕顏色不是底色，表示球碰到打板，改變 V 的值爲 Z 以加速 X 座標的變化。而 W 的值則根據球所碰到打板的位置來決定。愈靠近

−5	−3	−1	1	3	5

兩端其反彈角度愈大，接著跳至步驟(i)形成迴圈。（上面長方形，表示打板被分爲 6 段，各對應不同之值，以決定球反彈的方向）

(viii)若新的 X 座標大於 0，並且該位置之螢幕顏色不爲底色，表示已打到磚塊，將 V 變號使球方向改變。計算得分，若得分小於 720，則跳至步驟(i)形成迴圈。

當得分等於 720 表示磚塊全部均被打光了，此時將跳出迴圈。

(3)繪出所有方塊（DRAW FIELD）355 − 450行

此段副程式在螢幕上繪出八行交錯排列的方塊。其執行過程如下：

(i)先在螢幕上繪出藍色的底色（40×40）。

(ii)在20，22，24，26，28，30，32，34八行繪出淡綠色與橙色交互排列的磚塊，每一磚塊寬 1 高 2。

(4) 760－790 行：

當磚塊統統打光時，會執行此段程式，在螢幕上印出 CONGRATULATIONS， YOU WIN!

(5) 800－1090 行：

當五個球出界後遊戲結束，此段程式是當比賽結束後，根據得分高低給評語，分數愈高評語愈好。S 表分數，

(i) 0 ≤ S ＜ 60 BAD 差，

(ii) 60≤ S ＜ 200 GOOD 好，

(iii) 200 ≤ S ＜ 420 VERY GOOD 很好，

(iv) 420 ≤ S ＜ 720 EXCELLENT 優。

(6) 1100 － 1190 行：

此段程式在螢幕上顯示：

"PUSH BUTTON FOR NEW GAME， ESC KEY TO END"，表示要繼續玩的話，就壓一下 BUTTON 0 ，不然按入 " ESC " 結束遊戲。此段程式將比賽我績與目前最高分數比較（PEEK(789)+PEEK(790)*256)，若較高則存入最新較高得分。若沒有破最高分數則最高分數保留不變。

(7) 印出每行磚塊的分數（PRINT POINT SCALE ）1300－1390 行

此段副程式，在螢幕上針對八行磚塊顯示出打掉每一行一塊磚塊的分數。螢幕上顯示：

"POINTS PER BRICK → 1 2 3 4 5 6 7 8 "

愈靠右邊則分數愈高。位址（787 ，788 ）用來記錄是否第一次玩，當 PEEK(787)＝37， PEEK(788)＝38 表示不是第一次玩。若是第一次玩則分別將 37 與 38 放入兩位址，並將最高得分位址（789 ，790 ）淸除成零。最後在螢幕下端印出目前最

高記錄″HIGH SCORE″PEEK(789)+PEEK(790)*256。

(8)額外得分(BONUS)1500 — 1690 行

當磚塊統統打光，得分為滿分720 ，此時將會呼叫此副程式，加上額外得分。額外得分是根據還剩幾個球來決定，失球愈少額外得分也就愈高。

額外得分＝所剩的球數(L)* 50 。

(9) 2000 — 3020 行

此段程式為打磚塊程式起始部份，其執行步驟如下：

(i)檢查 PADDLE是否插好，本遊戲沒有PADDLE不能玩。

(ii)設定所有的變數，增加執行速度。(2140)

(iii)印出說明。(2340 — 2350)(2150 — 2240)

(iv)繪出所有方塊及印出每行磚塊得分。

(v)檢查是否使用 PADDLE0，本遊戲使用PADDLE0 來控制打板，PADDLE1 無效。

(vi)檢查五個球是否已經用完了，若是則印出得分。

(vii)將球數(L)減 1 ，使用RND隨機函數發球，開始比賽。

螢幕上所顯示的說明如下：

遊戲開始時螢幕上會顯示(2340 — 2350)

```
========================================

   L I T T L E   B R I C K   O U T

========================================

   COPYRIGHT 1979 APPLE COMPUTER INC
(CPRESSING THE ESC KEY WILL END THE GAME)
PRESS THE SPACE BAR TO BEGIN.........
```

按下SPACE BAR後螢幕顯示(2150 — 2240)

```
****************************************
* BRICK OUT IS A DYNAMIC BOARD GAME    *
* IN WHICH YOU ATTEMPT TO KNOCK DOWN   *
* A WALL OF BRICKS WITH A BALL         *
* STRUCK BY YOUR PADDLE.               *
*                                      *
*                                      *
*                                      *
* YOU CONTROL THE PADDLE ON THE BOARD  *
* BY TURNING THE APPLE PADDLE (0).     *
*                                      *
*                                      *
*                                      *
* YOU MAY 'PUT ENGLISH' ON THE BALL    *
* (DEFLECT THE BALL) BY HITTING IT     *
* WITH THE PADDLE HELD OFF-CENTER.     *
*                                      *
*                                      *
*                                      *
* BEFORE WE BEGIN, PLEASE TYPE         *
* YOUR FIRST NAME AND THEN PRESS       *
* THE KEY MARKED 'RETURN'              *
****************************************
```

(10) 4810 － 4830

此段副程式在螢幕上顯示

"<<<PLEASE USE THE OTHER PADDLE.>>>"

說明使用PADDLE 0 不要用PADDLE 1，PADDLE 1無效。

(11)發聲副程式（SOUND ROUTINE）7000 － 7030

此段副程式爲用組合語言所寫成的發聲副程式，其起始位置爲768。

(12)插入PADDLE （8500 － 8600）

在程式起始部份，會檢查PADDLE是否插入，若沒有插入將呼叫此副程式。本副程式執行時螢幕顯示：

" BRICK OUT REQUIRES THE USE OF THE APPLE "

″PADDLE SET PLEASE PLUG IN THE PADDLE″

″SET AND THEN PRESS THE SPACE BAR″

″IF YOU DO NOT HAVE PADDLES, YOU MAY″

″EXIT THE GAME BY PRESSING THE ESC KEY″

說明打磚塊一定要使用PADDLE，若遊戲者有paddle則應將其插上，否則按下″ESC″結束程式。

⒀按入″ESC″（11000－11110）

當主程式讀到″ESC″時，將執行本段副程式。螢幕上將顯示：

″WOULD YOU LIKE TO CONTINUE PLAYING″

″（TYPE YES OR NO AND PRESS RETURN）″

再次查詢是否眞的要停止遊戲，防止″ESC″不小心的鍵入。若回答″NO″則螢幕將顯示

″WOULD SOMEONE ELSE LIKE TO PLAY？″

若回答″YES″則遊戲將繼續下去。若回答″NO″則將結束遊戲。

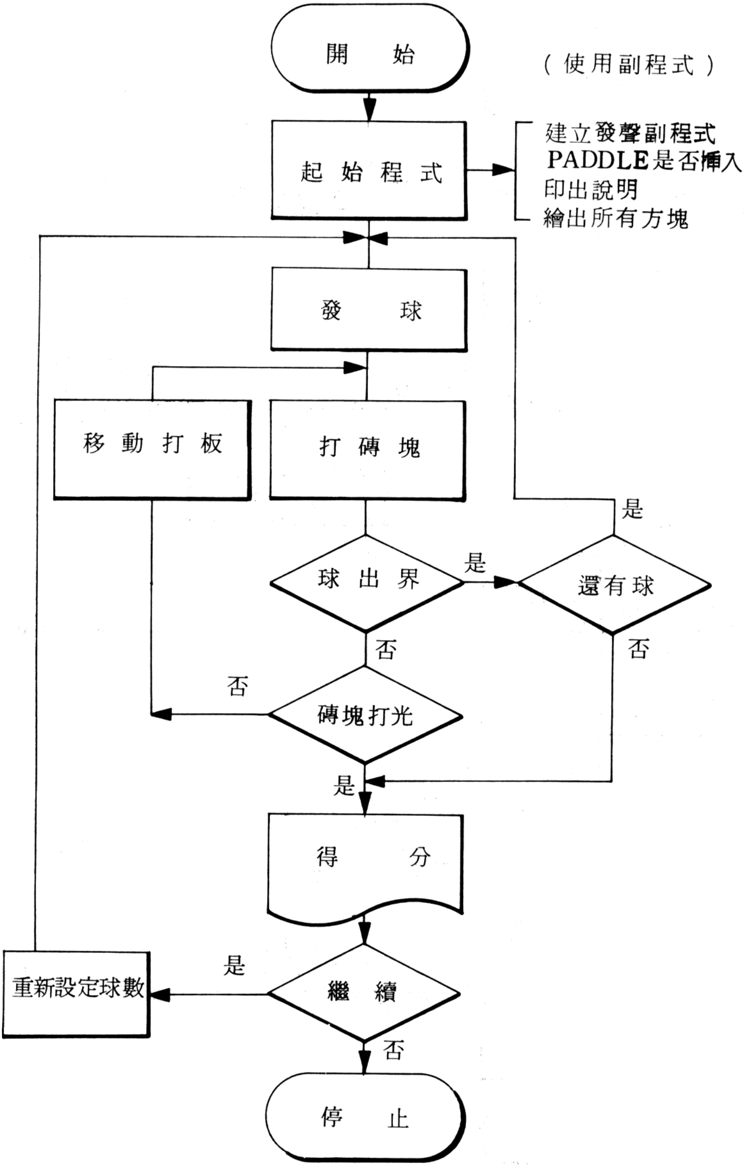
⒁31111（END）

結束程式的執行。

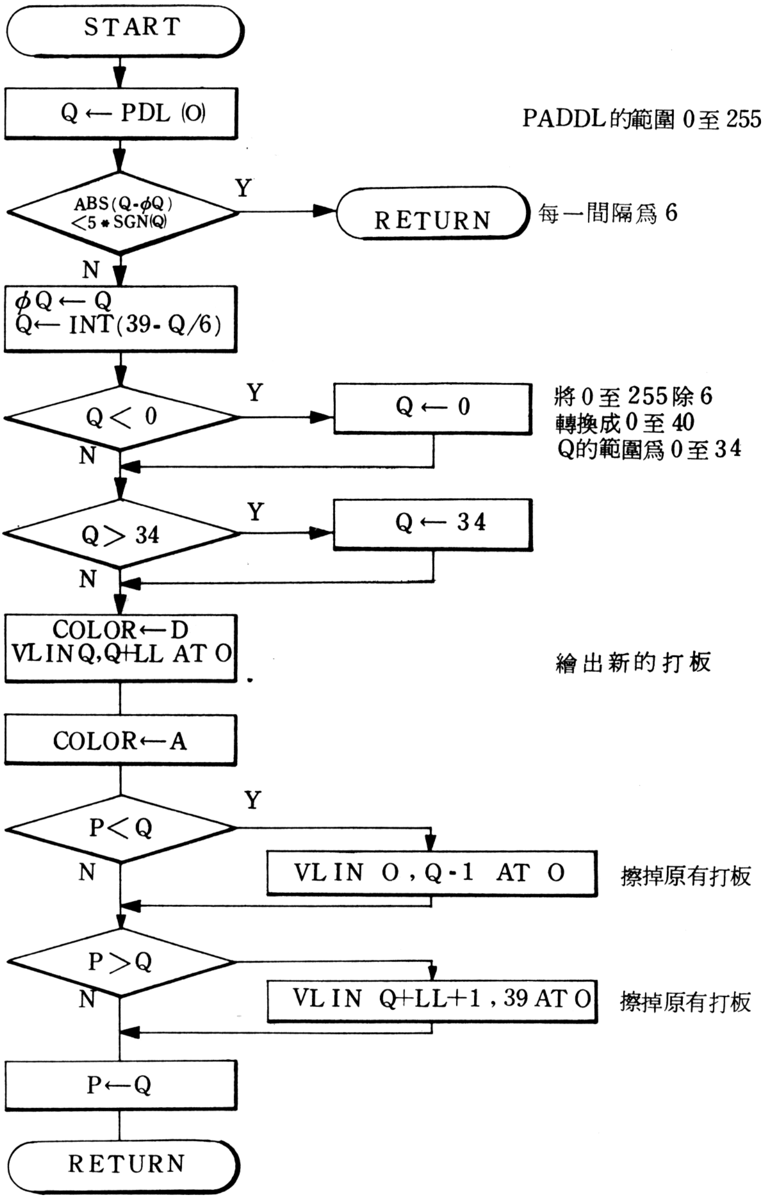
程式之流程圖

(1)主要流程

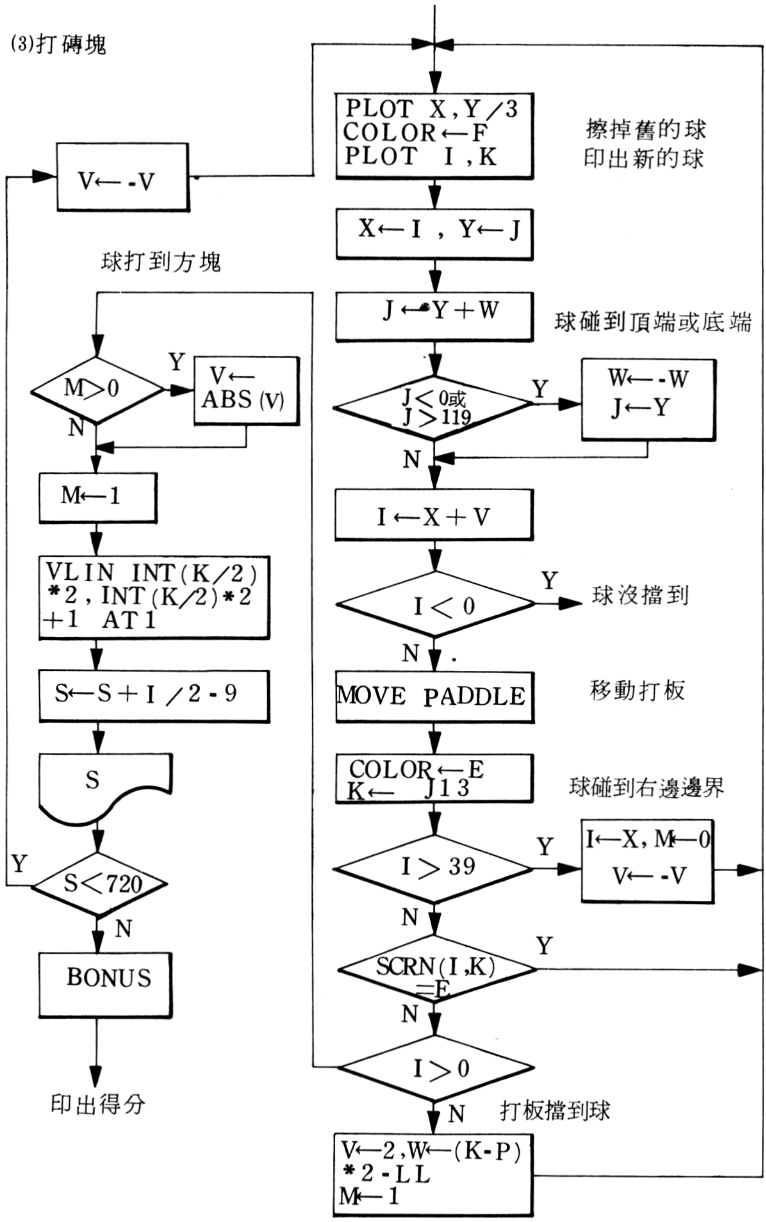
程式流程圖


開　始
（使用副程式）
起始程式
建立發聲副程式
PADDLE是否插入
印出說明
繪出所有方塊
發　球
移動打板
打磚塊
是
球出界
是
還有球
否
否
否
磚塊打光
是
得　分
是
重新設定球數
繼　續
否
停　止

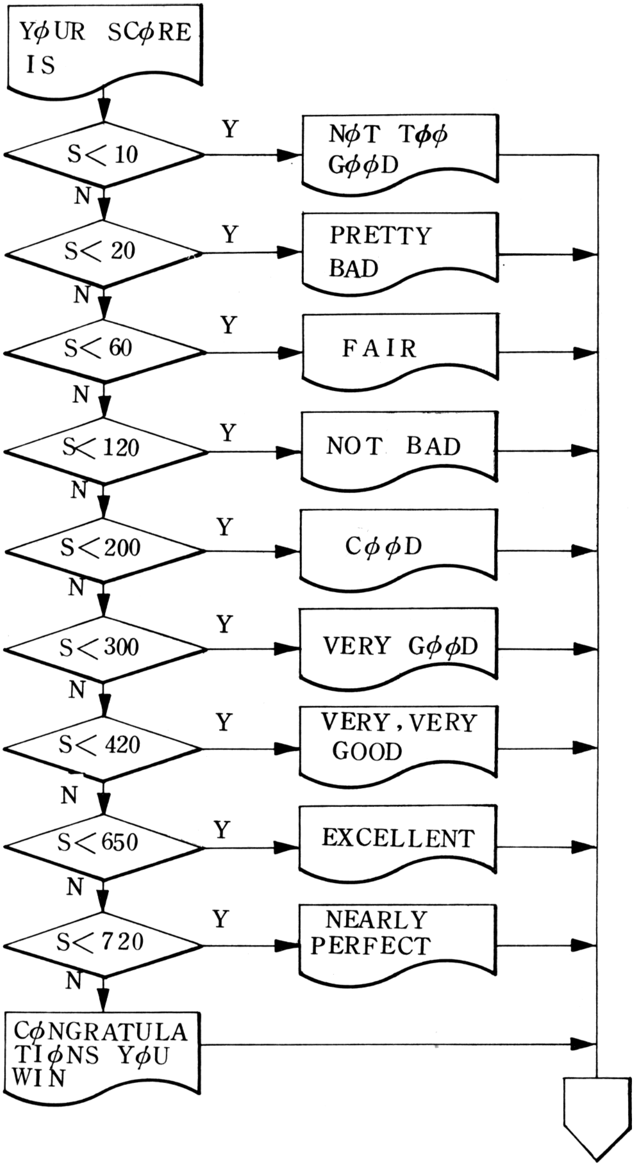
(2)移動打板


START
Q ← PDL (O)
PADDL的範圍 0至255
ABS(Q-φQ)
<5 * SGN(Q)
Y
RETURN
每一間隔爲 6
N
φQ ← Q
Q← INT(39- Q/6)
Q< 0
Y
Q ← 0
將0至255除6
轉換成0至40
Q的範圍爲0至34
N
Q> 34
Y
Q ← 34
N
COLOR←D
VLINQ,Q+LL AT O
繪出新的打板
COLOR←A
P<Q
Y
VLIN O , Q - 1 AT O
擦掉原有打板
N
P>Q
VLIN Q+LL+1 , 39 AT O
擦掉原有打板
N
P←Q
RETURN

(3)打磚塊


PLOT X,Y/3
COLOR←F
PLOT I,K
擦掉舊的球
印出新的球
X←I , Y←J
J←Y+W
球碰到頂端或底端
J<0或
J>119
Y
W←-W
J←Y
N
I←X+V
I<0
Y
球沒擋到
N
MOVE PADDLE
移動打板
COLOR←E
K←J13
球碰到右邊邊界
I>39
Y
I←X, M←0
V←-V
N
SCRN(I,K)
=E
Y
N
I>0
N
打板擋到球
V←2,W←(K-P)
*2-LL
M←1
V←-V
球打到方塊
M>0
Y
V←
ABS(V)
N
M←1
VLIN INT(K/2)
*2, INT(K/2)*2
+1 AT1
S←S+I/2-9
S
Y
S<720
N
BONUS
印出得分

(4)得分


YφUR SCφRE IS
S< 10
Y
NφT Tφφ GφφD
N
S< 20
Y
PRETTY BAD
N
S< 60
Y
FAIR
N
S< 120
Y
NOT BAD
N
S< 200
Y
CφφD
N
S< 300
Y
VERY GφφD
N
S< 420
Y
VERY, VERY GOOD
N
S< 650
Y
EXCELLENT
N
S< 720
Y
NEARLY PERFECT
N
CφNGRATULATIφNS YφU WIN

除了上述說明之外，程式中還用到一些技巧以製造一些效果：

(1)程式開始進行不久便跳到 2140 行執行，這一行看起來沒有任何作用，事實上這一行對程式的影響極爲重大。在第二章中作者已提過，在程式一開頭便將所有的變數設定，能使這些變數被放在 Applesoft 變數表（SYMBOL TABLE）的前面，往後程式中使用到這些變數時，APPLESOFT便能很快的從變數表中找出所需要的變數，節省不少搜尋的時間，而能增加程式的執行速度。

(2)程式中 2160 行：

″2160 COLOR ＝10：VLIN 1,46 AT 0:VLIN 0,47 AT 39″

剛看到這一行時可能會覺得很奇怪：前面並沒有設定爲 GR，且執行後也沒有在指定位置出現顏色碼爲10的直線。事實上本行的作用是在螢幕上第 0 及39列（COLUMN）顯示由″*″星號組成的二條直線，配合程式中的程式說明使用。由於此行前面並沒有宣告 GR 故此畫線動作是在文字幕上進行，又因COLOR＝10，故會將文字幕第 0 及 39列所對應的記憶區位址存入″AA″（因一個位址的 8 個位元，分爲二半分別對應低解像螢幕的上、下二個畫圖點，詳細情形請見 1 、 3 二章）而在文字幕上顯示出″*″（圖″*″的 ASCII 碼即爲 $AA）。這種方式雖然很怪，卻不失爲一個很方便的方法。

(3) 2250行的 POKE PEEK(40)+PEEK(41)*256+PEEK(36),96 。

此行的目的便是在螢幕上目前游標所在位置，存入96之值，以便在螢幕上出現一閃爍的空白字形，即產生一個游標。同理程

式中 2450行執行後便為：

PRESS THE SPACE BAR TO BEGIN.. ▩→閃爍的游標

否則此文句後便不會出現一游標。

(4)程式中利用位址 789，790 存放目前遊戲的最高分，而不像一般程式利用變數來存放最高分數。利用這種方式最大的好處便是縱然此程式停此執行後，只要不關機或改變此位址之內容，往後再執行此程式時程式仍能知道以前遊戲者的最高分。這種方式在一般遊戲程式中常常被用到。

(5)程式中第 215行的作用在產生出不同的聲音。因球碰到打板的不同位置時，變量W之值也不相同。故存入位址 6 當作聲音高低的值便不相同，再呼叫768之發聲程式便能發出高低不同的聲音。因此整個遊戲給遊戲者的感覺便好像是球碰到打板的不同位置便發出不同的反彈聲。

結語

此程式因為是使用BASIC 所寫成的程式（較不具結構性），並且使用到APPLE Ⅱ 特殊的記憶位址，因此不太容易看懂。但若仔細看其結構技巧，就會發覺這個程式並不太難懂。這個程式使用到 PADDLE，低解析度螢幕顯示圖形，和特定螢幕畫面控制，此外還有一個發聲副程式，因此讀者若有心在 APPLE Ⅱ 上寫 GAME，祇要能體會程式中的技巧，即可寫出各種五光十色含有音效的程式來，發揮 APPLE Ⅱ 的所有功能。

最後在結束本文之前須提示讀者一點，即程式中對於球及打板的移動，是先移動球再移動打板。這種方式使得遊戲者較不容易擋到球，因此有些遊戲者常會覺得〃奇怪！已經轉動了 PADDLE，怎麼打板沒有動？害我沒擋到。〃。事實上對於這一點

我們可以對程式稍加修改，便可改掉這個〃缺點〃，至於如何修改呢？只好請讀者試試看了！

```
]LIST

10  GOTO 2000
20 Q =  PDL (0): IF  ABS (Q - QQ)
      < 5 *  SGN (Q) THEN  RETURN

21 Z =  PEEK ( - 16384): IF Z = 1
     55 THEN 11010
30 QQ = Q:Q =  INT (39 - Q / 6): IF
     Q < 0 THEN Q = 0
40  IF Q > 34 THEN Q = 34
50  COLOR= D: VLIN Q,Q + LL AT 0:
      COLOR= A: IF P < Q THEN  VLIN
     0,Q - 1 AT 0
60  IF Q < P THEN  VLIN Q + LL +
     1,39 AT 0
70 P = Q: RETURN
75  PLOT X,Y / 3: COLOR= F: PLOT
     I,K:X = I:Y = J
80 J = Y + W: IF J < 0 OR J > 119
      THEN  GOSUB 230
120 I = X + V: IF I < 0 THEN 1900

130  IF I < 6 THEN QQ = 500
150  GOSUB 20: COLOR= E:K = J / 3
     : IF I > 39 THEN 240
160  IF  SCRN( I,K) = E THEN 75
170  IF I THEN 270
180 V = 2:W = (K - P) * 2 - LL:M =
     1
215  POKE 6,150 -  ABS (W) * 4: POKE
     7,3
218 Z =  PEEK ( - 16336) -  PEEK
     ( - 16336) +  PEEK ( - 16336
     ) -  PEEK ( - 16336): CALL 7
     68
220  GOTO 75
230 W =  - W:J = Y:Z =  PEEK ( -
     16336) -  PEEK ( - 16336) +
      PEEK ( - 16336) -  PEEK ( -
```

```
       16336) +  PEEK ( - 16336) -
        PEEK ( - 16336) +  PEEK ( -
       16336): RETURN : REM  SIDE B
       OUNCE
240    FOR I = 1 TO 6:M =  PEEK ( -
       16336): NEXT I:I = X:M = 0
250 V =  - V
260    GOTO 75
270    IF M THEN V =  ABS (V)
275 M = 1
280    VLIN  INT (K / 2) * 2, INT (
       K / 2) * 2 + 1 AT I:S = S +
       I / 2 - 9: VTAB 24: HTAB 8: PRINT
       S;: VTAB 23: HTAB 1
295    POKE 6,180 - I * 3: POKE 7,3
       :Z =  PEEK ( - 16336): CALL
       768
300    POKE 6,140 + I *  RND (1): POKE
       7,2: CALL 768
310    POKE 6,50: POKE 7,3: CALL 76
       8
330    REM ^BRICK STRUCK
340    IF S < 720 THEN 250
350    VTAB 24: PRINT : PRINT : PRINT
       : GOSUB 1500: GOTO 760
355    COLOR= E: REM DRAW FIELD
360    FOR I = 0 TO 39: VLIN 0,39 AT
       I: NEXT I
420    FOR I = 20 TO 34 STEP 2
430    COLOR= B: VLIN 0,39 AT I: COLOR=
       C: FOR W =  INT ((I / 4 -  INT
       (I / 4)) * 4 + .05) *  SGN (
       I / 4) TO 39 STEP 4
440    VLIN W,W + 1 AT I: NEXT W
450    NEXT I: RETURN
760    VTAB 23: PRINT "CONGRATULATI
       ONS, ";: IF  LEN (A$) THEN  PRINT
       A$;", ";
770    PRINT "YOU WIN!";: CALL  - 8
       68: PRINT
780 A$ = "CHAMP"
790    GOTO 1100
800    VTAB 24: PRINT : PRINT : PRINT
       : VTAB 22: PRINT "YOUR SCORE
        IS ";: IF S < 10 THEN 910
810    IF S > 50 THEN EXTP = 1
```

```
820  IF S > 100 THEN NVC = 0
830  IF S < 20 THEN 960
840  IF S < 60 THEN 1010
850  IF S < 120 THEN 1040
860  IF S < 200 THEN 1050
870  IF S < 300 THEN 1060
880  IF S < 420 THEN 1070
890  IF S < 650 THEN 1080
900  GOTO 1090
910  IF NVC THEN 930
920  PRINT "NOT TOO GOOD, BUT": PRINT
     "KEEP ON TRYING!":NVC = 1: GOTO
     1100
930  IF NVC1 THEN 950
940  PRINT "STILL NOT GREAT,": PRINT
     "BUT KEEP WORKING AT IT.":NV
     C1 = 1: GOTO 1100
950  PRINT "ALL RIGHT.": PRINT "J
     UST KEEP TRYING.": GOTO 1100

960  IF NVC THEN 990
970  IF  NOT EXTP THEN 910
980  PRINT "PRETTY BAD!": GOTO 11
     00
990  PRINT "MUCH BETTER THAN IT W
     AS.": PRINT "KEEP GOING!": GOTO
     1100
1000  PRINT "O.K.": PRINT "TRY IT
      AGAIN!": GOTO 1100
1010  IF NVC THEN 1030
1020  PRINT "FAIR. ": GOTO 1100
1030  PRINT "GETTING REALLY GOOD.
     ": GOTO 1100
1040  PRINT "NOT BAD. ": GOTO 110
     0
1050  PRINT "GOOD.": GOTO 1100
1060  PRINT "VERY GOOD.": GOTO 11
     00
1070  PRINT "VERY, VERY GOOD.": GOTO
     1100
1080  PRINT "EXCELLENT.": GOTO 11
     00
1090  PRINT "NEARLY PERFECT.": GOTO
     1100
1100  POKE 50,63: VTAB 24: PRINT
     "PUSH BUTTON FOR NEW GAME, E
```

```
          BONUS POINTS:           "
1520  IF L > 1 THEN  PRINT "BONUS
      POINTS FOR ";L;" BALLS LEFT
     :           "
1530  VTAB 22: FOR Q = 1 TO 1800:
      NEXT Q
1540  FOR Q = 0 TO L * 50 STEP 5
1550  HTAB 32: PRINT Q;
1560  POKE 6,180: POKE 7,10: CALL
     768
1570  POKE 6,200 *  RND (1) + 50:
      POKE 7,10: CALL 768
1580  POKE 6,160 - Q / 10: POKE 7
     ,12: CALL 768
1590  NEXT Q: POKE 50,255
1600 BANUS = Q - 5: VTAB 21: HTAB
     1: PRINT "TOTAL=";
1610  FOR Q = 1 TO 500: NEXT Q
1620  FOR Q = S TO S + BANUS STEP
     5
1630  HTAB 8: PRINT Q;
1640  POKE 6,150 - Q / 40: POKE 7
     ,3: CALL 768
1650  POKE 6,140: POKE 7,2: CALL
     768
1660  POKE 6,50: POKE 7,3: CALL 7
     68
1680  FOR K = 1 TO 10 *  RND (1) +
     20: NEXT K
1690  NEXT Q: PRINT : RETURN
1900  FOR H = 1 TO 80:OD =  PEEK
     ( - 16336): NEXT H: GOTO 291
     0
2000  REM  *** START OF PROGRAM *
     **
2010  GOSUB 7000: REM  POKE SOUND
      ROUTINE
2020  TEXT : CALL  - 936: POKE  -
     16298,0
2090  IF  PEEK ( - 16287) > 127 THEN
      IF  PEEK ( - 16286) > 127 THEN
      GOSUB 8500
2100  REM  ^IF BOTH PADDLE SWITCH
     ES READ "ON", THEN PADDLES A
     RE NOT PLUGGED IN.
2110  VTAB 24
```

```
     SC KEY TO EN";: POKE 2039,4:
      POKE 50,255: POKE  - 16368,
     0
1110  REM  "POKE 2039,4" POKES AN
      INVERSE "D" INTO THE LAST S
     CREEN POSITION, AVOIDING SCR
     OLLING.
1120 S = S + BANUS:BANUS = 0
1130  IF S <  PEEK (789) +  PEEK
     (790) * 256 THEN 1160: REM
     NEW SCORE LESS THAN HIGH SCO
     RE
1140  POKE 790,S / 256: POKE 789,
     S -  PEEK (790) * 256: REM
     POKE NEW HIGH SCORE INTO PLA
     CE
1150  VTAB 21: HTAB 15: PRINT "FO
     RMER"
1160 Z =  PEEK ( - 16384): IF  PEEK
     ( - 16287) > 127 THEN 1190
1170  GOSUB 20: IF Z <  > 155 THEN
     1160
1175  REM
1180  IF Z = 155 THEN 11000
1190 OQ = 500: GOTO 2520
1300  REM  PRINT POINT SCALE
1310 SCOLE = 6 - LL
1320  CALL  - 936: PRINT "POINTS
     PER BRICK -> 1 2 3 4 5 6 7 8
     "
1360  VTAB 24: CALL  - 958: PRINT
     "SCORE= 0";
1370  IF  PEEK (787) = 37 THEN  IF
      PEEK (788) = 38 THEN 1390: REM
      IF "VARIABLE" ALREADY INITI
     ALIZED, SKIP
1380  POKE 787,37: POKE 788,38: POKE
     789,0: POKE 790,0: REM  SET
     FLAGS AND INIT NON-CLEARING
     VARIABLE
1390  HTAB 22: PRINT "HIGH SCORE=
      "; PEEK (789) +  PEEK (790)
      * 256;: HTAB 1: RETURN
1500  REM  BONUS
1510  VTAB 22: POKE 50,63: IF L =
     1 THEN  PRINT "
```

```
2140 Q = I = LL = J = P = OQ = X =
     Y = K = W = E = V = D = F =
     N = M = Z = S: REM SET UP VA
     RIABLE TABLE FOR MAXIMUM SPE
     ED
2145  GOSUB 2340
2150  PRINT "********************
     *******************";: VTAB
     1: HTAB 1: PRINT "**********
     ****************************
     *"
2160  COLOR= 10: VLIN 1,46 AT 0: VLIN
     0,47 AT 39: REM  "DRAW" ASTE
     RICKS (*) ALONG EDGES
2170  VTAB 3: PRINT "* BRICK OUT
     IS A DYNAMIC BOARD GAME": PRINT
     "* IN WHICH YOU ATTEMPT TO K
     NOCK DOWN"
2180  PRINT "* A WALL OF BRICKS W
     ITH A BALL": PRINT "* STRUCK
      BY YOUR PADDLE.": PRINT : PRINT

2190  PRINT "* YOU CONTROL THE PA
     DDLE ON THE BOARD": PRINT "*
      BY TURNING THE APPLE PADDLE
      (O).": PRINT : PRINT
2200  PRINT "* YOU MAY "; CHR$ (3
     4);"PUT ENGLISH"; CHR$ (34);
     " ON THE BALL": PRINT "* (DE
     FLECT THE BALL) BY HITTING I
     T": PRINT "* WITH THE PADDLE
      HELD OFF-CENTER."
2210  PRINT : PRINT
2220  POKE  - 16368,0
2240  PRINT "* BEFORE WE BEGIN, P
     LEASE TYPE": PRINT "* YOUR F
     IRST NAME AND THEN PRESS": PRINT
     "* THE KEY MARKED "; CHR$ (3
     4);"RETURN"; CHR$ (34);
2250  POKE  PEEK (40) +  PEEK (41
     ) * 256 +  PEEK (36),96: REM
      CURSOR
2260 P =  PEEK ( - 16384): IF P =
     155 THEN 11000
2270  IF  PEEK ( - 16287) > 127 THEN
     2335
```

```
2280  IF P < 156 AND P <  > 141 THEN
     2260
2290  INPUT "";A$
2300  IF  LEN (A$) = 0 THEN 2335
2310  FOR I = 1 TO  LEN (A$): IF
      ASC ( MID$ (A$,I,I)) <  > 6
     4 THEN 2330
2320  NEXT I:I = I - 1
2330 A$ =  MID$ (A$,I, LEN (A$)):
      IF  LEN (A$) THEN  IF  LEN
     (A$) > 12 THEN A$ =  LEFT$ (
     A$,12)
2335  GOTO 2490
2340  CALL  - 936: VTAB 5: PRINT
     "=============================
     ==============";: INVERSE : PRINT
     "   L I T T L E   B R I C K
       O U T       ";: NORMAL : PRINT
     "=============================
     =============="
2350  VTAB 12: PRINT "   COPYRIGH
     T 1979 APPLE COMPUTER INC"
2410  VTAB 20
2420  PRINT : PRINT "(PRESSING TH
     E ESC KEY WILL END THE GAME)
     "
2450  VTAB 23: PRINT "PRESS THE S
     PACE BAR TO BEGIN....";: POKE
      PEEK (40) +  PEEK (41) * 25
     6 +  PEEK (36),96: POKE  - 1
     6368,0
2460 P =  PEEK ( - 16384): IF  PEEK
     ( - 16287) > 127 THEN 2480
2470  IF P <  > 160 AND P <  > 15
     5 THEN 2460
2480  POKE  - 16368,0: IF P = 155
      THEN 11000
2485  HOME : VTAB 24: HTAB 1: RETURN

2490  CALL  - 936
2500  GR :A = 2:B = 12:C = 9:D =
     13:E = 2:F = 15: REM COLORS
2510 LL = 5:P = 15
2520  GOSUB 355
2580 OOLDP1 =  PDL (1)
2860  GOSUB 1300
```

```
      8: POKE 50,63: PRINT "  PUSH
       PADDLE BUTTON TO BEGIN THE
      GAME  ";: POKE 50,255:OO = 5
      00: RETURN
4810   VTAB 23: HTAB 1: CALL  - 86
      8: PRINT " <<< PLEASE USE TH
      E OTHER PADDLE. >>>  ";
4820   POKE 6,40: POKE 7,30: CALL
      768
4830  ERASE = 20: RETURN
7000   REM  SOUND ROUTINE
7010   POKE 768,173: POKE 769,48: POKE
      770,192: POKE 771,136: POKE
      772,208: POKE 773,4: POKE 77
      4,198: POKE 775,7: POKE 776,
      240: POKE 777,8:
7020   POKE 778,202: POKE 779,208:
       POKE 780,246: POKE 781,166:
       POKE 782,6: POKE 783,76: POKE
      784,0: POKE 785,3: POKE 786,
      96
7030   RETURN
8500   REM  PADDLES NOT PLUGGED IN

8510   VTAB 10: PRINT "BRICK OUT R
      EQUIRES THE USE OF THE APPLE
      "
8520   PRINT "PADDLE SET.  PLEASE
      PLUG IN THE PADDLE"
8530   PRINT "SET AND THEN PRESS T
      HE SPACE BAR."
8540   PRINT : PRINT "IF YOU DO NO
      T HAVE PADDLES, YOU MAY"
8550   PRINT "EXIT THE GAME BY PRE
      SSING THE ESC KEY."
8560  P =  PEEK ( - 16384): IF P <
      155 THEN 8560
8570   POKE  - 16368,0: IF P <  >
      155 THEN 8580: GOTO 11000
8580   IF  PEEK ( - 16287) > 127 AND
       PEEK ( - 16286) > 127 THEN
      8560
8590   IF P <  > 160 THEN 8560
8600   CALL  - 936: TEXT : RETURN
      : REM  PADDLES ARE IN AND SP
      ACE BAR HAS BEEN PRESSED.
```

```
2870 S = 0:Y = 19:L = 6
2880  VTAB 23: POKE 50,63: PRINT
     "  PUSH PADDLE BUTTON TO BEG
     IN THE GAME  ";: POKE 50,255

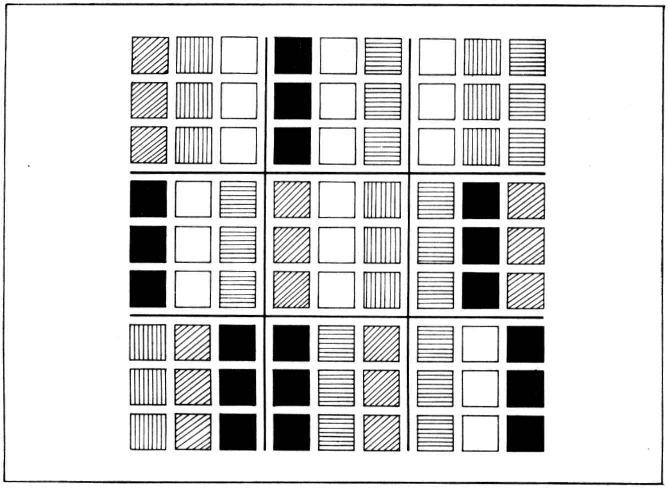
2890  FOR K = 1 TO 20: NEXT K: REM
      ALLOW TIME FOR USER TO RELE
     ASE BUTTON FROM LAST PUSH
2900  GOSUB 20
2904 P1 =  PDL (1): IF ( ABS (OOL
     DP1 - P1) > 3 OR  PEEK ( - 1
     6286) > 127) AND  NOT ERASE THEN
      GOSUB 4810
2905 OOLDP1 = P1: REM WRONG PADDL
     E
2906  IF ERASE THEN ERASE = ERASE
      - 1
2907  IF ERASE = 1 THEN  GOSUB 48
     00
2909  IF  PEEK ( - 16287) < 128 AND
      PEEK ( - 16384) <  > 155 THEN
     2900
2910  VTAB 23: CALL  - 868
2920  COLOR= A: PLOT X,Y / 3:Y =
     120 *  RND (1)
2930 X = 19
2940 V =  - 1:W = 5 *  RND (1) -
     2:L = L - 1: IF L < 1 THEN 8
     00
2947  IF  LEN (A$) THEN  IF L > 1
      THEN  PRINT A$", ";
2950  IF L > 1 THEN  PRINT "THIS
     IS BALL ";6 - L;" OUT OF 5."

2970  IF  PEEK ( - 16384) = 155 THEN
     11000
2980  IF L <  > 1 THEN 3010
2990  POKE 50,63: PRINT "THIS IS
     THE LAST BALL";
3000  POKE 50,255: VTAB 23: PRINT

3010  FOR I = 1 TO 25 + 125 *  RND
     (1): GOSUB 20: NEXT I:M = 1:
     N = 0
3020  GOTO 80
4800  VTAB 23: HTAB 1: CALL  - 86
```

```
11000  REM  "ESC" KEY HAS BEEN PR
     ESSED
11010  POKE  - 16368,0: CALL  - 9
     36: IF  PEEK (34) = 0 THEN  VTAB
     10
11015  PRINT "WOULD YOU LIKE TO C
     ONTINUE"
11020  PRINT "PLAYING? ";:VTB =  PEEK
     (37):TB =  PEEK (36)
11030  PRINT : PRINT : PRINT "(TY
     PE YES OR NO AND PRESS RETUR
     N)";
11040  VTAB VTB + 2: CALL  - 868:
      VTAB VTB + 1: POKE 36,TB: CALL
      - 868: INPUT "";H$
11050  IF  LEFT$ (H$,1) <  > "Y" THEN
     11060
11055 H$ = "":P = 15:OQ = 500: GOTO
     2490
11060  IF  LEFT$ (H$,1) <  > "N" THEN
     11040
11070  VTAB (VTB): CALL  - 868: VTAB
     (VTB + 1): PRINT "WOULD SOME
     ONE ELSE LIKE TO PLAY? ";:VT
     B =  PEEK (37):TB =  PEEK (3
     6)
11080  VTAB 23: CALL  - 868: VTAB
     VTB + 1: POKE 36,TB: CALL  -
     868: INPUT "";H$
11090  IF  LEFT$ (H$,1) = "Y" THEN
     2020
11100  IF  LEFT$ (H$,1) <  > "N" THEN
     11080
11110  GOTO 31111
31111  POKE  - 16368,0: TEXT : CALL
      - 936: TEXT : END
55000  IF  INT (39 -  PDL (0) / 6
     ) > 34 THEN P = 34
```

程式八　魔術方塊 (MAD. MAD CUBE)



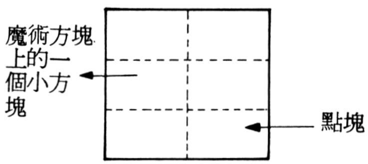
你一定看過前一陣子流行的立體魔術方塊，非常的吸引人，一旦你迷上它便很難放的下手。魔術方塊由27個塑膠方塊所組成，形成一個實心六方體，因此它可以沿三個座標軸旋轉。當你買到一個新的魔術方塊時，它的六個面有著不同的顏色——紅、藍、綠、棕、橘、黃。然後可以旋轉任意面上的 9 個小方塊，使得六個面上不同的顏色混在一起。此時若要將各個面上的顏色，恢復成原來單一的顏色將是一件很不容易的事。當你在旋轉方塊使它還原的過程中，若移動方向錯誤，將使顏色更加混亂。但儘管

玩魔術方塊是一件累人的事，人們還是花無數的時間去破解它。

魔術方塊的遊戲程式，可以說是另一種類型的遊戲程式，並不太難，跟其他幾種遊戲程式在觀念上不太相同，而且本程式提示我們一些新的想法和技巧。只要讀者參照本書說明，研讀後面所列的程式，將可輕而易舉，事半功倍，相信您一定可領悟到您自己的東西。以下我們便正式開始介紹，在APPLE Ⅱ 上模擬魔術方塊的過程！

立方體的表示

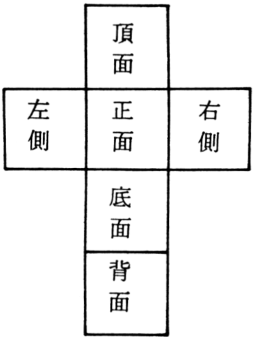
第一個問題就是如何在計算機上顯示一個立方體。首先考慮魔術方塊各面上的小方塊，在 APPLE 中應如何顯示。我們知道 APPLE Ⅱ 的低解像圖面的每個小點塊（即低解像圖形的一個點），並非正方形而是成一橫長方形，長比寬約長一半。因此程式中使用一 3 × 2 的小點塊來表示出魔術方塊上的一個小方塊。



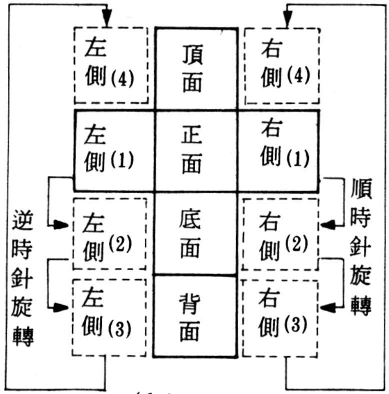
此 3 × 2 的點塊在螢幕上顯示後便很像正方形了。接著便須決定，在同一方向上螢幕可放幾個魔術方塊面。魔術方塊共有六個面，每個面有 9 個小方塊，橫向縱向各 3 個，而在垂直方向每個小方塊需用 3 個點表示，共需9 個點塊。但每個方塊間需有間隔，因此共需 3 × 4 ＝12 個點塊，才可以表示出一垂直方向的一個魔術方塊的一個面的長度。接著我們要計算的便是，螢幕上是否能容納四個面排成一列，4 個面乘以12個點點總共需要 48個點塊。APPLE Ⅱ 低解析度螢幕容量是48×40，垂直方向剛好可以容納四個面，不過此時螢幕底部的四列文字列將無法使用（對於全螢幕所使用到的指令與特定位址可查閱 APPLE

REFERENCE MANUAL APPENDIX B ）。

且在程式中，採用平面圖形表示出魔術方塊的立體圖形，因此在低解像圖形顯示時，必須將立方體展開，依上面討論的方式平攤在螢幕上，其排列方式如下圖(a)：



（a）魔術方塊之六面



（b）

此時螢幕上左、右兩側還有空位。對於兩側的空位，程式中將其分別放在原左側與右側的面各經旋轉後所得到的圖形，如上圖(b)，（圖中所指的順、反時針旋轉方向為程式中所採用的方向，操作遊戲時並不會使用到，但在追踪程式時便須注意了），亦即加上的 6 個面，分別為當頂面、底面、背面為正面時其左右兩個側面，如此在螢幕上總共顯示十二個面。

程式的操作方式

魔術方塊可以沿著 3 個座標軸旋轉，每次可選擇位於同一平面上的 9 個方塊旋轉，可依順時針或逆時針方向逆轉，旋轉度數則為90的任意倍數。但因此時整個低解像螢幕均用來顯示魔術方

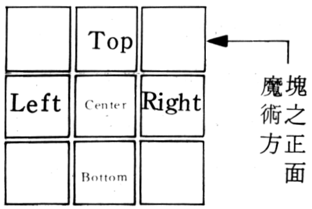
塊的各個不同的面，原來位於低解像螢幕底部的四列文字列無法使用，因此需使操作的指令欲簡單愈好，並儘量不去使用此四列文字列。

首先旋轉必定是90度的倍數，因此不論要旋轉幾度均可以用連續旋轉90度來完成。眞正的魔術方塊爲立體（三維），但在螢幕上顯示的僅爲平面（二維），因而程式在設計上使用平面的二個座標軸來改變正面，第三個軸（垂直於平面者）可以用平面的軸連續旋轉三次來改變其方向。

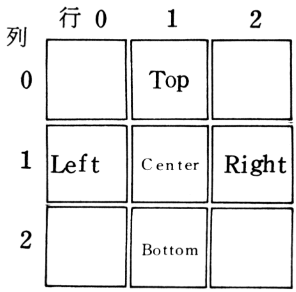
至於輸入方面本程式是採用 JOYSTICK（搖桿）及 push - button（按鈕）。此搖桿好比位於正面中央，若要旋轉時，只須將此搖桿推至所欲旋轉的方向後，按一下按鈕，便可告訴計算機去執行所要的旋轉。

操作的方式有兩種：一爲旋轉同一平面上的任意 9 個小方塊，一爲改變正面。

(一)旋轉方塊的過程包括兩個步驟，首先選定正面上欲旋轉的行或列，可利用搖桿來選擇。其方式爲：程式首先閃動自定的搖桿所在位置即左圖中的 Center 方塊，接著按讀入的搖桿位置，閃動搖桿所指示的欲旋轉的行或列，讀者應注意到一個面上雖有 9 個小方塊，但只有圖中所示的五個方塊爲搖桿所指示的有效位置。若程式閃動 Center 的方塊後，搖桿向上推則 Center 之方塊會停止閃動，而變成Top 方塊閃動，此時若再將搖桿向右推，則又變爲 Right 方塊閃動（並非Top方塊的右邊方塊閃動），因爲此時搖桿所指示的是欲旋轉的整行或整列，Center ，Top 、Left 、Right、Bottom



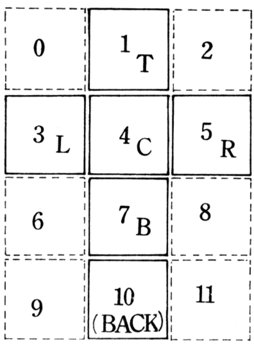
之外的四個方塊無法指出到底是那一行或那一列的旋轉。待選定欲旋轉的行或列後，按下PUSHBUTTON ＃ 0 ，小方塊會停止閃動，然後便可推動搖桿朝所要旋轉的方向推去，即可完成旋轉。



如左圖：

1. Top 表示要旋轉第 0 列，可向左、右旋轉。
2. Left 表示要旋轉第 0 行，可向上、下旋轉。
3. Right 表示要旋轉第 2 行，可向上、下旋轉。
4. Bottom 表示要旋轉第 2 列，可向左、右旋轉。
5. Center 表示要旋轉第 1 行或第 1 列，由上、下、左、右旋轉來決定。

㈡旋轉正面的步驟則是先按下PUSHBUTTON ＃ 1 後，再將搖桿推向所要改變正面的方向。如圖左，想將面 5 變成正面則將搖桿推向 L 向左轉；欲將面 7 變成正面則可將搖桿推向 T 使原來的面 7 變成正面，原來的面 4 移成面 1 ，面 3 與面 5 的位置不變，但均旋轉了90度。此外讀者應注意圖左各面的代號，為程式中所使用的代號，讀者應注意其所對應的各面。



保存你的魔術方塊：

由於魔術方塊很不容易破解，故程式中特別為有磁碟機的讀者加上可將目前的魔術方塊圖形存於磁碟上的能力，如此程式

可隨時停止或繼續自未完成的狀態再開始。

可能讀者沒有搖桿或覺得搖桿沒有鍵盤方便，可修改程式中的輸入指令處理部份，使得輸入改成用鍵盤：先輸入C，L，R，T，B然後輸入方向L，R，U，D。但筆者認爲原程式的搖桿處理方式，可做爲學習讀入搖桿對應值的很好範例，讀者應徹底了解。

此外立方體兩側經旋轉後輸出的六個面，若你認爲太複雜，亦可予以刪除。當然，本書所列出的程式除了希望讀者徹底了解外，更希望讀者能按照自己的意思，再多加修改，使程式更加完美。

程式中主要變數說明

(1) NC（11，2，2）：由於APPLESOFT 的陣列註標由0算起，故所宣告的長度較實際使用時小1。NC表示New Color，用來記錄所顯示的12個面與每個面上的3行3列中的小方塊的顏色，即12面×3行×3列故宣告成三維陣列NC（0:11，0:2，0:2）其中第1註標表示面，第2註標表示行。第3註標表示列。

(2) HC（2，2）：HC表示Hold Color，用來暫存魔術方塊12個面中某一個面上各小方塊的顏色，使NC的改變能順利進行。

(3) X，Y：爲螢幕座標，X爲橫座標（0至39），Y爲縱座標（0至47）。

(4) FA，RO，CO：FA 代表 FACE；RO 代表ROW；CO

代表COLUMN。

(5) NF，OF　　：NF 表示 New Face；OF 表示 old Face。

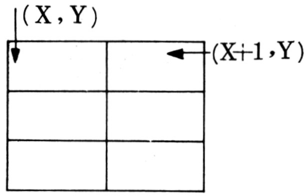
(6) PO$　　：PO$ 表示 Position 即搖桿所指示的位置其值爲'C'，'L'，'R'，'T'，'B'五者之一。

(7) COLOR　　：BK ＝ 0 爲 BLACK；RD ＝ 1 爲 RED；BL ＝ 2 爲 BLUE；GN ＝ 4 爲 GREEN；BR ＝ 8 爲 BROWN；NG ＝ 9 爲 ORANGE；YE ＝ 13 爲 YELLOW；WH ＝ 15 爲 WHITE 均在程式開始時設定好。

副程式說明

(1) DRAW SOUARE AT （X，Y）：30 － 50 行

利用迴圈在座標（X，Y），畫一高 3 寬 2 的低解像點小方塊。



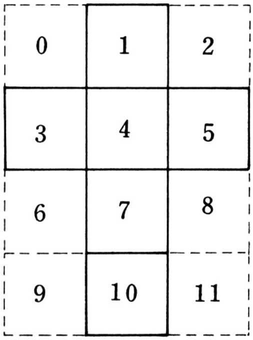
(2) DRAW CUBE ON SCREEN ：70 － 160 行

利用三層迴圈將 NC（11，2，2）所代表的12個面，按各面方塊的顏色均顯示在螢幕上（各方塊間有空隔）。

(3) DO EQUIVALENT ROTATES ：180 － 400 行

下圖中 1、3、4、5、7、10 六個面是魔術方塊展開後所得到的六個面。其餘六個面中 6、9、10 三個面是經由面 3 連續

反時針旋轉所得；而 8、11、2 則是由面 5 連續順時針旋轉所得。換個角度來看，多出的六個面分別是面 1、7、10為正面時的左、右兩個側面。



(3 － 1) ROTATE COUNTER：
190 － 250 行
此段程式將一個面逆時針方向旋轉90度。

(3 － 2) ROTATE CLOCKWISE ：260 － 320 行
此段程式將一個面順時針方向旋轉90度。

上面（ 3 － 1 ）、（ 3 － 2 ）兩個副程式主要功用在於產生一個面的順時針或逆時針旋轉面，供(3)中之 340 － 390 行呼叫。如：

面 3 爲

Y	Y	Y
G	G	G
R	R	R

則面 6 爲

Y	G	R
Y	G	R
Y	G	R

面 9 爲

R	R	R
G	G	G
Y	Y	Y

面 0 爲

R	G	Y
R	G	Y
R	G	Y

如

面 5 爲

B	B	B
W	W	W
O	O	O

則面 8

O	W	B
O	W	B
O	W	B

面 11

O	O	O
W	W	W
B	B	B

面 2

B	W	O
B	W	O
B	W	O

(4) MOVE COLUMN UP ： 1530 － 1590 行

將正面 4 的第 0 、 1 或 2 行向上旋轉90度。假設所旋轉的行數爲CO←{ 0 ， 1 ， 2 } 。其步驟如下：

(i)將面 1 暫存在 HC（ 0：2，0：2 ）中。

(ii)將 CO行由面 4 抄至面 1 。(iii)將CO行由面 7 抄至面 4 。

(iv)將 CO行由面10抄至面 7 。(v)將 HC 抄至面10。

如此便可將特定的行數（CO）向上旋轉90度。

在本副程式與下一個副程式中，使用到下列副程式：

(a) 1670 行將面 FA 抄至 HC 中暫存

(b) 1680 行將 HC 的 CO 行抄至面 FA 的 CO 行。

(c) 1690 行將 OF （OLD FACE）的 CO 行抄至 NF （NEW FACE）的 CO 行。

(5) MOVE COLUMN DOWN ： 1600 － 1660 行

將正面 4 的第 0 、1 或 2 行向下旋轉90度。假設所要旋轉的行數為 CO ←{0，1，2}，則步驟如下：

(i)將面10暫存至 HC（0：2，0：2）中。

(ii)將 CO 行由面 7 抄至面 10。(iii)將CO行由面 4 抄至面 7 。

(iv)將 CO 行由面 1 抄至面 4 。(v)將 HC 抄至面 1 。

如此可將特定的行數（CO）向下旋轉90度。

(6) MOVE ROW RIGHT ： 1700 － 1760 行

將正面 4 的第 0 、1 或 2 列向右旋轉90度。假設所要旋轉的列數為 RO ←{0，1，2}，則步驟如下：

(i)將面 4 暫存於 HC （Hold Color）中。

(ii)將 RO 列由面 3 抄至面 4 。(iii)將RO列由面10抄至面 3 。

(iv)將 RO 列由面 5 抄至面10 。(v)將 HC 抄至面 5 。

如此可將特定的ROW向右旋轉90度。

在本副程式與下個副程式(7)中，呼叫下列副程式：

(a) 1840 行將面 FA 抄至 HC 中暫存。

(b) 1850 行將 HC 之 RO 列抄至面 FA 的 RO 列。

(C) 1860 行將面 4 的 RO 列抄至面 3 或面 5 的 RO 列。

(d) 1870 行將面10 的 RO 列抄至面 3 或面 5 的 RO 列。

(e) 1880 行將面 3 或面 5 的RO 列抄至面10的RO 列。

(7) MOVE ROW LEFT ： 1770 — 1830 行

將正面 4 的第 0 、 1 或 2 列向左旋轉90度。假設所要旋轉列數是 RO ← { 0 ， 1 ， 2 } ，其步驟如下：

(i)將面 4 暫存於 HC 中。

(ii)將 RO 列由面 5 抄至面 4 。(iii)將RO列由面10 抄至面 5 。

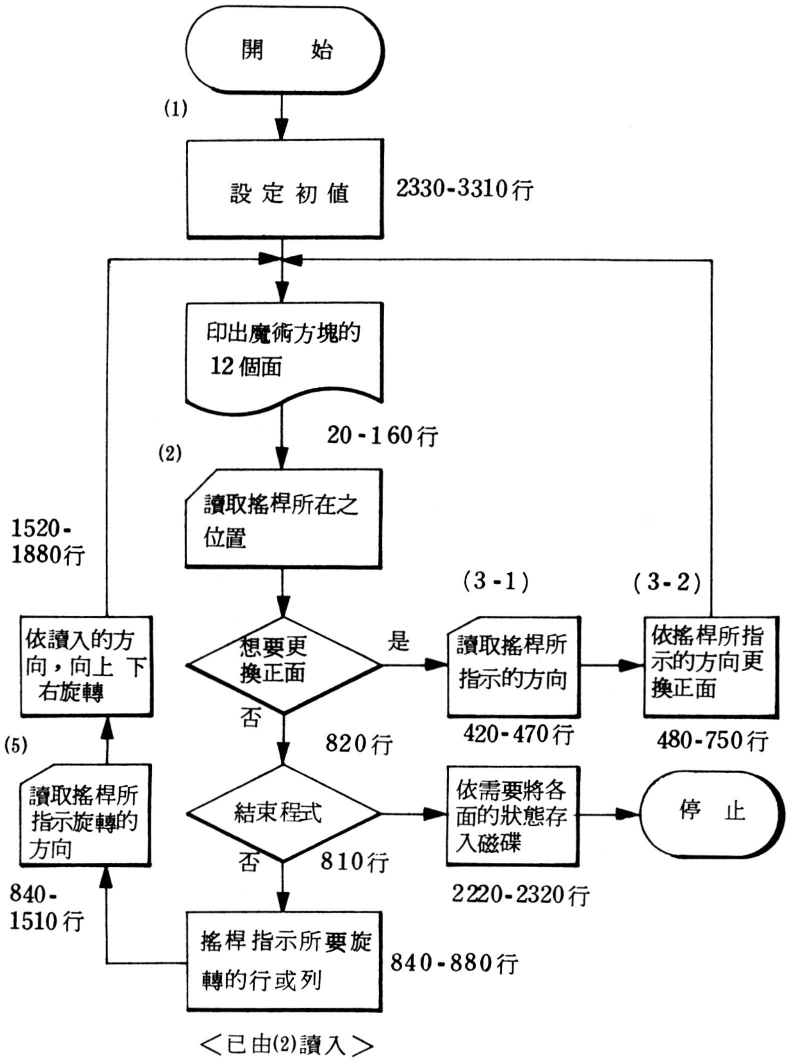
(iv)將 RO 列由面 3 抄至面10 。(v)將 HC 抄至面 3 。

如此可將特定的ROW向左旋轉90度。

此外由於立體結構，面10的 RO 列爲原來之（ 2 —RO）列，即面10與面 3 ， 4 ， 5 的方向剛好相反。（ 1870 ， 1880 行）

程式流程說明

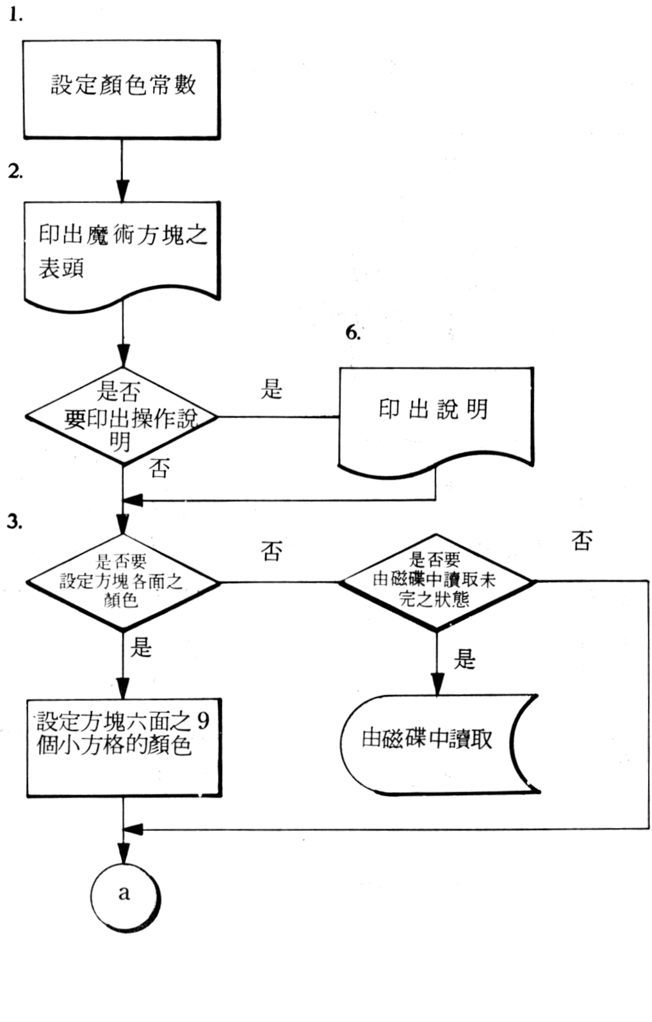
以下我們將對程式的流程利用流程圖做很詳細的分析，讀者可以發現，藉著本說明可很輕鬆的追踪此程式。流程一爲程式進行的主要流程，對於另需說明的部份，在各流程單元的左上角有號碼代表說明的次序。

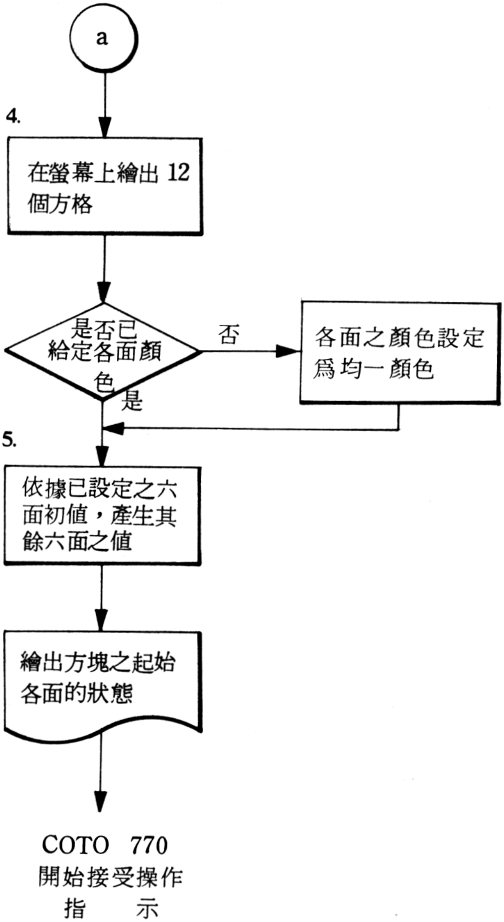


流程一：程式之主要流程
（流程單元中左上角有數碼者
另有分解說明）

<1>設定初值：2330 — 3310 行

我們可再以流程二表示出此部份的流程：




a
4.
在螢幕上繪出 12 個方格
是否已給定各面顏色
否
是
各面之顏色設定爲均一顏色
5.
依據已設定之六面初值，產生其餘六面之值
繪出方塊之起始各面的狀態
COTO 770
開始接受操作指　示

1.設定顏色常數： 2370 行

與 APPLE Ⅱ 中GRAPHIC MODE相同 BK ＝0 黑；RD ＝1 紅；BL ＝2 藍；GN ＝4 綠；GR ＝8 棕；NG＝ ＝9 橘；Y E ＝13黃；WH ＝15白。

2.印出表頭：〃＊＊＊THE MAD MAD MAD MAD CUBE ＊＊＊〃

3.處理方塊各面顏色初值的設定

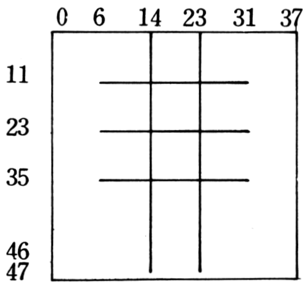
初值設定有下列三種方式：

(i)對每面的 9 個小方塊一個一個設定 2420 － 2650 行。

(ii)從磁碟中讀取以前存在磁碟中魔術方塊的顏色分配情形，即繼續先前未解出的狀態。 2740 － 2800 行。

(iii)六個面各面的 9 個方塊給予相同的顏色：面 3 紅色；面 1 藍色；面 5 黃色；面 4 綠色；面 7 橘色；面10白色。 2810 － 2870 行。

4.在螢幕上繪出12個方格。 2680 － 2710 行。



繪出棕色的五條線做爲12個面的邊界，如左圖所示。

5.依據六面初值產生其餘六面，由面 1 ，3 ，4 ，5 ，7 ，10可產生面 0 ，2 ，6 ，8 ，9 ，11；呼叫 DO EQUIVALENT ROTATES 副程式得之。

6.程式執行前所顯示的說明： 2900 － 3310 行

程式開始前，會詢問使用者是否需要操作說明，若要，便會在螢幕上顯示出三頁的說明文字（利用 space bar 換頁），藉此使使用者了解如何輸入指令。

<2> 讀取搖桿所在的位置

BLINK JOYSTICK POSITION ： 1970 — 2200 行

由搖桿指示出所要旋轉的行或列。

(a)首先要決定搖桿的位置，程式中自定之值為 PO$ = ″C″ center ，然後根據搖桿之值來改變 PO$。

(i) PDL(0)<50 & PDL(1)>50　PO$=″L″　LEFT

(ii) PDL(0)>200 & PDL(1)>50　PO$=″R″　RIGHT

(iii) PDL(1)<50 & PDL(0)>50　PO$=″T″　TOP

(iv) PDL(1)>200 & PDL(0)>50　PO$=″B″　BOTTOM

(b)接著要決定行與列，其設定之值為ROW = 1 ，COLUMN=1

(i) PO$=″L″時，

ROW= 1 ，COLUMN = 0

(ii) PO$=″R″時，

ROW = 0 ，COLUMN = 2

(iii) PO$=″T″時，

ROW= 0 ，COLUMN=1

(iv) PO$=″B″時，ROW = 2 ，COLUMN=1

（ROW, COLUMN）

	(0,1) T	
(1,0) L	(1,1) C	(1,2) R
	(2,1) B	

(c)最後根據ROW與COLUMN之值在該小方塊中，先顯示黑色再顯示原來的顏色，如此便可使該小方塊達到閃動的效果，並完成搖桿所在位置的讀取。

<3> 更換正面

(3-1) ROTATE AXIS ： 420 － 470 行

更換正面主要先由搖桿決定改變的方向

(i)PDL(1)>200 向下，面 1 變爲正面。(ii)PDL(1)<50向上，面 7 變成正面 (iii)PDL(0)> 200 向右，面 3 變成正面 (iv) PDL(0)<50向左，面 5 成爲正面。

(3-2) 1. ROTATE AXIS LEFT ： 490 － 540 行，正面向左轉，包括 5 個步驟：

(i)面 4 第 0 列向左轉90度。(ii)面 1 順時針旋轉90度 (iii)面 4 第 1 、 2 列向左轉90度 (iv) 面 7 逆時針旋轉90度(v)面 0 、 2 、 6 、 8 、 9 、11 也要由新的面 3 與面 5 產生。

如此便可將正面由原來的面 4 換成面 5 。此外須注意旋轉的動作是呼叫前面所介紹過的副程式來完成。

2. ROTATE AXIS RIGHT ： 550 － 610 行，正面向右轉包括 5 個步驟：

(i)面 4 第 0 列向右轉90度 (ii)面 1 逆時針旋轉90度 (iii)面 4 第 1 、 2 列向右轉90 度 (iv)面 7 順時針旋轉90度(v)面 0 、 2 、 6 、 8 、 9、11也要由新的面 3 與面 5 產生。

如此可將正面由原來的面 4 換成面 3 。

3. ROTATE AIXS DOWN ： 620 － 680 行，正面向下轉包含 5 個步驟：

(i)面 4 第 0 行向下轉90度(ii) 將面 0 抄至面 3 (iii) 面 4 第 1 、 2 行向下轉90度(iv) 將面 2 抄至面 5 (v)

面0、2、6、8、9、11由新的面3與面5產生。

如此可將正面由原來的4換成1。

4. ROTATES AXIS UP ： 690 − 750行，正面向上轉包含5個步驟：

(i)面4第0行向上轉90度(ii)將面6抄至面3(iii)面4第1、2行向上轉90度(iv)將面8抄至面5(v)面0、2、6、8、9、11由新的面3與面5產生。

如此可將正面由原來的面4換成面7。

<4>結束程式之處理： 2220 − 2320 行

在程式開始執行後，在Keyboard上按下任意一個鍵，便表示你要結束程式的執行，此時程式會詢問你是否要將目前的魔術方塊顏色記錄下來，若回答要，程式便會將NC存入磁碟上。

這一部份程式牽涉到DOS的一些基本指令，沒有磁碟機的讀者可予以略過。了解DOS的讀者可作爲參考，相當容易。

<5>依指令旋轉方塊

PLAY CUBE 840 − 1510行

此處我們已由搖桿知道所要旋轉的行與列，接著我們需知道其旋轉的方向，而根據搖桿之值：

(i) PDL(1)>200 ，向下　　(ii)PDL(1)<50 ，向上

(iii)PDL(0)>200 ，向右　　(iv)PDL(0)<50 ，向左

便可知道旋轉的方向。

其旋轉共有12種方式：

(1) LEFT COLUMN MOVE DOWN ： 940 − 960行

第 0 行向下旋轉90度，其步驟如下：

(i)第 0 行向下旋轉90度。(ii) 面 3 順時針旋轉90度。(iii) 面 3 逆時針旋轉，以產生新的面 0 、6 、9 。

(2) LEFT COLUMN MOVE UP ： 970 － 990 行

第 0 行向上旋轉90度，其步驟如下：

(i)第 0 行向上旋轉90度。(ii) 面 3 逆時針旋轉90度。(iii) 面 3 逆時針旋轉，以產生新的面 0 、6 、9 。

(3) RIGHT COLUMN MOVE DOWN ： 1050 － 1070 行

第 2 行向下旋轉90度，其步驟如下：

(i)第 2 行向下旋轉90度。(ii) 面 5 逆時針旋轉90度。(iii) 面 5 順時針旋轉，以產生新的面 2 、8 、11。

(4) RIGHT COLUMN MOVE UP ： 1080 － 1100 行

第 2 行向下旋轉90度，其步驟如下：

(i)第 2 行向上旋轉90度。(ii) 面 5 順時針旋轉90度。(iii) 面 5 順時針旋轉，以產生新的面 2 、8 、11。

(5) TOP ROW MOVE RIGHT ： 1160 － 1180 行

第 0 列向右旋轉90度，其步驟如下：

(i)第 0 列向右旋轉90度。(ii) 面 1 逆時針旋轉90度。(iii) 面 0 、2 、6 、8 、9 、11由面 3 與面 5 旋轉產生。

(6) TOP ROW MOVE LEFT ： 1190 － 1210 行

第 0 列向左旋轉90度，其步驟如下：

(i)第 0 列向左旋轉90度。(ii) 面 1 順時針旋轉90度。(iii) 面 0 、2 、6 、8 、9 、11 由面 3 與面 5 旋轉產生。

(7) BOTTOM ROW MOVE RIGHT ： 1270 － 1290 行

第 2 列向右旋轉90度，其步驟如下：

(i)第 2 列向右旋轉90度。(ii) 面 7 順時針旋轉90度。(iii)

面0、2、6、8、9、11由面3與面5旋轉產生。

(8) BOTTOM ROW MOVE LEFT ： 1300 － 1320 行

第2列向左旋轉90度，其步驟如下：

(i)第2列向左旋轉90度。(ii)面7逆時針旋轉90度。(iii)面0、2、6、8、9、11由面3與面5旋轉產生。

(9) CENTER ROW MOVE RIGHT ： 1400 － 1420 行

第一列向右旋轉90度，其步驟如下：

(i)第一列向右旋轉90度。

(ii)面0、2、6、8、9、11由面3及面5重新旋轉產生。

(10) CENTER ROW MOVE LEFT ： 1430 － 1450 行

第一列向左旋轉90度，其步驟如下：

(i)第一列向左旋轉90度。

(ii)面0、2、6、8、9、11由面3及面5重新旋轉產生。

(11) CENTER COLUMN MOVE DOWN ： 1460 － 1480 行

第一行向下旋轉90度，其步驟如下：

(i)第一行向下旋轉90度。

(ii)面0、2、6、8、9、11由面3與面5重新旋轉產生。

(12) CENTER COLUMN MOVE UP ： 1490 － 1510 行

第一行向上旋轉90度，其步驟如下：

(i)第一行向上旋轉90度。

(ii)面0、2、6、8、9、11由面3與面5重新旋轉產生。

在上面的12種不同的旋轉方式，使用到副程式：

①向上旋轉90度　　②向下旋轉90度

③向左旋轉90度　　④向右旋轉90度

⑤產生兩側的順、反時針旋轉面。

各副程式在前面均已說明。

整個程式到此已全部說明完畢，由於魔術方塊是立體結構，一部份的旋轉，會引起其他部份的改變，一切都必須考慮清楚，但整個程式說起來，只要弄清楚程式中對魔術方塊的表示方法與操作方式，並不會太難。

```
]LIST

1  REM  ***************************
   *****
2  REM  ** THE MAD MAD MAD MAD CU
   BE **
3  REM  **     BY RICH HATCHER
        **
4  REM  **   COPYRIGHT (C) 1980 B
   Y   **
5  REM  **     MICRO-SPARC, · INC.
        **
6  REM  **     LINCOLN, MA 01773
        **
7  REM  ***************************
   *****
10  GOTO 2340
20  REM  ********************
30  REM  DRAW SQUARE AT X,Y
40  FOR W = X TO X + 1: VLIN Y,Y +
    2 AT W: NEXT W
50  RETURN
60  REM  ********************
70  REM  DRAW BOXES ON SCREEN
80  FOR FA = 0 TO 11
90  FOR RO = 0 TO 2
100  FOR CO = 0 TO 2
110 X = (FA -  INT (FA / 3) * 3) *
     9 + CO * 3 + 6
```

```
120 Y =  INT (FA / 3) * 12 + RO *
     4
130  COLOR= NC(FA,RO,CO)
140  GOSUB 30
150  NEXT CO: NEXT RO: NEXT FA
160  RETURN
170  REM  ********************
180  REM  DO EQUIVALENT ROTATES
190  REM  ROT CTR CLOCKWISE
200  FOR RO = 0 TO 2: FOR CO = 0 TO
     2:HC(RO,CO) = NC(OF,RO,CO): NEXT
     CO: NEXT RO
210  FOR NR = 0 TO 2: FOR MC = 0 TO
     2
220 DR = MC:OC = 2 - NR
230 NC(NF,NR,MC) = HC(DR,OC)
240  NEXT MC: NEXT NR
250  RETURN
260  REM  ROT CLOCKWISE
270  FOR RO = 0 TO 2: FOR CO = 0 TO
     2:HC(RO,CO) = NC(OF,RO,CO): NEXT
     CO: NEXT RO
280  FOR NR = 0 TO 2: FOR MC = 0 TO
     2
290 DR = 2 - MC:OC = NR
300 NC(NF,NR,MC) = HC(DR,OC)
310  NEXT MC: NEXT NR
320  RETURN
330  REM  FACES 0,2,6,8,9,11 ARE
     ROTATE COPY OF 3,5
340 NF = 6:OF = 3: GOSUB 200
350 NF = 8:OF = 5: GOSUB 270
360 NF = 9:OF = 6: GOSUB 200
370 NF = 11:OF = 8: GOSUB 270
380 NF = 0:OF = 9: GOSUB 200
390 NF = 2:OF = 11: GOSUB 270
400  RETURN
410  REM  ********************
420  REM  ROTATE AXIS
430  IF  PDL (1) > 200 THEN 620
440  IF  PDL (1) < 50 THEN 690
450  IF  PDL (0) > 200 THEN 550
460  IF  PDL (0) < 50 THEN 480
470  GOTO 430
480  REM  ROTATES AXIS LEFT
490 RO = 0: GOSUB 1770
```

```
500 NF = 1:OF = 1: GOSUB 270
510 RO = 1: GOSUB 1770
520 RO = 2: GOSUB 1770
530 NF = 7:OF = 7: GOSUB 200
540  GOTO 1950
550  REM  ROTATES AXIS RIGHT
560 RO = 0: GOSUB 1700
570 NF = 1:OF = 1: GOSUB 200
580 RO = 0: GOSUB 1700
590 RO = 2: GOSUB 1700
600 NF = 7:OF = 7: GOSUB 270
610  GOTO 1950
620  REM  ROTATES AXIS DOWN
630 CO = 0: GOSUB 1600
640 NF = 3:OF = 0: GOSUB 1890
650 CO = 1: GOSUB 1600
660 CO = 2: GOSUB 1600
670 NF = 5:OF = 2: GOSUB 1890
680  GOTO 1950
690  REM  ROTATES AXIS UP
700 CO = 0: GOSUB 1530
710 NF = 3:OF = 6: GOSUB 1890
720 CO = 1: GOSUB 1530
730 CO = 2: GOSUB 1530
740 NF = 5:OF = 8: GOSUB 1890
750  GOTO 1950
760  REM  ********************
770  REM  INPUT MOVE
780  CALL  - 198: REM  BELL
790  GOSUB 1970: REM  BLINK
800 X =  PEEK ( - 16287): IF X >
     128 THEN 840: REM  PLAY
810 Y =  PEEK ( - 16384): IF Y >
     128 THEN 2220: REM  END
820 Z =  PEEK ( - 16286): IF Z >
     128 THEN 420: REM  FULL ROTA
     TES
830  GOTO 790
840  IF PO$ = "L" THEN 900
850  IF PO$ = "R" THEN 1010
860  IF PO$ = "T" THEN 1120
870  IF PO$ = "B" THEN 1230
880  IF PO$ = "C" THEN 1340
890  REM  ********************
900  REM  LEFT COLUMN MOVE
910  IF  PDL (1) > 200 THEN 950
```

```
920  IF  PDL (1) < 50 THEN 980
930  GOTO 910
940  REM  MOVE DOWN
950 CO = 0: GOSUB 1600
960 NF = 3:OF = 0: GOSUB 1890: GOTO
     1950
970  REM  MOVE UP
980 CO = 0: GOSUB 1530
990 NF = 3:OF = 6: GOSUB 1890: GOTO
     1950
1000  REM  ********************
1010  REM  RIGHT COLUMN MOVE
1020  IF  PDL (1) > 200 THEN 1060

1030  IF  PDL (1) < 50 THEN 1090
1040  GOTO 1020
1050  REM  MOVE DOWN
1060 CO = 0: GOSUB 1600
1070 NF = 5:OF = 2: GOSUB 1890: GOTO
     1950
1080  REM  MOVE UP
1090 CO = 2: GOSUB 1530
1100 NF = 5:OF = 8: GOSUB 1890: GOTO
     1950
1110  REM  ********************
1120  REM  TOP ROW MOVE
1130  IF  PDL (0) > 200 THEN 1170

1140  IF  PDL (0) < 50 THEN 1200
1150  GOTO 1130
1160  REM  MOVE RIGHT
1170 RO = 0: GOSUB 1700
1180 NF = 1:OF = 1: GOSUB 200: GOTO
     1950
1190  REM  MOVE LEFT
1200 RO = 0: GOSUB 1770
1210 NF = 1:OF = 1: GOSUB 270: GOTO
     1950
1220  REM  ********************
1230  REM  BOTTOM ROW MOVE
1240  IF  PDL (0) > 200 THEN 1280

1250  IF  PDL (0) < 50 THEN 1310
1260  GOTO 1240
1270  REM  MOVE RIGHT
1280 RO = 2: GOSUB 1700
```

```
1290 NF = 7:OF = 7: GOSUB 270: GOTO
     1950
1300  REM  MOVE LEFT
1310 RO = 2: GOSUB 1770
1320 NF = 7:OF = 7: GOSUB 200: GOTO
     1950
1330  REM  ********************
1340  REM  CENTER ROW OR COLUMN M
     OVE
1350  IF  PDL (0) > 200 THEN 1410

1360  IF  PDL (0) < 50 THEN 1440
1370  IF  PDL (1) > 200 THEN 1470

1380  IF  PDL (1) < 50 THEN 1500
1390  GOTO 1350
1400  REM  MOVE RIGHT
1410 RO = 1: GOSUB 1700
1420  GOTO 1950
1430  REM  MOVE LEFT
1440 RO = 1: GOSUB 1770
1450  GOTO 1950
1460  REM  MOVE DOWN
1470 CO = 1: GOSUB 1600
1480  GOTO 1950
1490  REM  MOVE UP
1500 CO = 1: GOSUB 1530
1510  GOTO 1950
1520  REM  ********************
1530  REM  MOVE CO UP
1540 FA = 1: GOSUB 1670
1550 NF = 1:OF = 4: GOSUB 1690
1560 NF = 4:OF = 7: GOSUB 1690
1570 NF = 7:OF = 10: GOSUB 1690
1580 FA = 10: GOSUB 1680
1590  RETURN
1600  REM  MOVE CO DOWN
1610 FA = 10: GOSUB 1670
1620 NF = 10:OF = 7: GOSUB 1690
1630 NF = 7:OF = 4: GOSUB 1690
1640 NF = 4:OF = 1: GOSUB 1690
1650 FA = 1: GOSUB 1680
1660  RETURN
1670  FOR RO = 0 TO 2:HC(RO,CO) =
     NC(FA,RO,CO): NEXT RO: RETURN
```

```
1680  FOR RO = 0 TO 2:NC(FA,RO,CO
     ) = HC(RO,CO): NEXT RO: RETURN

1690  FOR RO = 0 TO 2:NC(NF,RO,CO
     ) = NC(OF,RO,CO): NEXT RO: RETURN

1700  REM  MOVE ROW RIGHT
1710 FA = 4: GOSUB 1840
1720 NF = 4:OF = 3: GOSUB 1860
1730 NF = 3:OF = 10: GOSUB 1870
1740 NF = 10:OF = 5: GOSUB 1880
1750 FA = 5: GOSUB 1850
1760  RETURN
1770  REM  MOVE ROW LEFT
1780 FA = 4: GOSUB 1840
1790 NF = 4:OF = 5: GOSUB 1860
1800 NF = 5:OF = 10: GOSUB 1870
1810 NF = 10:OF = 3: GOSUB 1880
1820 FA = 3: GOSUB 1850
1830  RETURN
1840  FOR CO = 0 TO 2:HC(RO,CO) =
     NC(FA,RO,CO): NEXT CO: RETURN

1850  FOR CO = 0 TO 2:NC(FA,RO,CO
     ) = HC(RO,CO): NEXT CO: RETURN

1860  FOR CO = 0 TO 2:NC(NF,RO,CO
     ) = NC(OF,RO,CO): NEXT CO: RETURN

1870  FOR CO = 0 TO 2:NC(NF,RO,CO
     ) = NC(OF,2 - RO,2 - CO): NEXT
     CO: RETURN
1880  FOR CO = 0 TO 2:NC(NF,2 - R
     O,CO) = NC(OF,RO,2 - CO): NEXT
     CO: RETURN
1890  REM  COPY FACE OF TO NF
1900  FOR RO = 0 TO 2
1910  FOR CO = 0 TO 2
1920 NC(NF,RO,CO) = NC(OF,RO,CO)
1930  NEXT CO: NEXT RO
1940  RETURN
1950  GOSUB 340: GOSUB 70
1960  GOTO 770
1970  REM  BLINK JOYSTICK POSITIO
     N
```

```
1980 PO$ = "C"
1990  REM  READ JOYSTICK
2000  IF  PDL (0) < 50 AND  PDL (
     1) > 50 THEN PO$ = "L"
2010  IF  PDL (0) > 200 AND  PDL
     (1) > 50 THEN PO$ = "R"
2020  IF  PDL (1) < 50 AND  PDL (
     0) > 50 THEN PO$ = "T"
2030  IF  PDL (1) > 200 AND  PDL
     (0) > 50 THEN PO$ = "B"
2040 RO = 1:CO = 1
2050  IF PO$ = "L" THEN CO = 0
2060  IF PO$ = "R" THEN CO = 2
2070  IF PO$ = "T" THEN RO = 0
2080  IF PO$ = "B" THEN RO = 2
2090 HC(RO,CO) = NC(4,RO,CO):NC(4
     ,RO,CO) = BK
2100  GOSUB 2140
2110 NC(4,RO,CO) = HC(RO,CO)
2120  GOSUB 2140
2130  RETURN
2140  REM  DRAW SQUARE AT FA,RO,C
     O
2150 FA = 4
2160 X = (FA -  INT (FA / 3) * 3)
      * 9 + CO * 3 + 6
2170 Y =  INT (FA / 3) * 12 + RO *
     4
2180  COLOR= NC(FA,RO,CO)
2190  GOSUB 30
2200  RETURN
2210  REM  ********************
2220  REM  TERMINATION PROCEDURE
2230  POKE  - 16368,0
2240  TEXT : HOME : PRINT "DO YOU
      WANT TO SAVE THIS GAME ?": GET
     Z$: PRINT Z$: IF Z$ = "Y" THEN
     2270
2250  PRINT : PRINT : PRINT : PRINT
     "DO YOU WANT TO CONTINUE PLA
     YING ?": GET Z$: PRINT Z$: IF
     Z$ = "N" THEN  END
2260 X$ = "Y": GOTO 2680
2270 D$ = CHR(4): PRINT D$"OPEN C
     OLOR CUBE OLD GAME"
2280  PRINT D$"WRITE COLOR CUBE O
```

```
    LD GAME"
2290 FA = 3: GOSUB 2320:FA = 1: GOSUB
    2320:FA = 5: GOSUB 2320:FA =
    4: GOSUB 2320:FA = 7: GOSUB
    2320:FA = 10: GOSUB 2320
2300  PRINT D$"CLOSE COLOR CUBE O
    LD GAME"
2310  GOTO 2250
2320  FOR RO = 0 TO 2: FOR CO = 0
     TO 2: PRINT NC(FA,RO,CO): NEXT
    CO: NEXT RO: RETURN
2330  REM  ********************
2340  REM  INITIALIZATION
2350  REM
2360  REM  COLOR ASSIGNMENT
2370 BK = 0:RD = 1:BL = 2:GN = 4:
    BR = 8:NG = 9:YE = 13:WH = 1
    5
2380  DIM NC(11,2,2),HC(2,2)
2390  HOME : VTAB 10: HTAB 5: INVERSE
    : PRINT "*** THE MAD MAD MAD
     MAD CUBE ***": NORMAL : HTAB
    15: PRINT "BY RICH HATCHER":
     PRINT : HTAB 10: PRINT "COP
    YRIGHT (C) 1981 BY": HTAB 10
    : PRINT "MICRO-SPARC, INC.
    ": HTAB 10: PRINT " LINCOLN,
     MA 01773"
2400  PRINT : PRINT "DO YOU WANT
    INSTRUCTION ?": GET Y$: PRINT
    Y$: IF Y$ = "Y" THEN  GOSUB
    2900
2410  HOME : PRINT "DO YOU WANT T
    O SET UP ?": GET X$: PRINT X
    $: IF X$ = "N" THEN 2660
2420  HOME : PRINT "  RED      =R
    ": PRINT "  BLUE      =B   ":
     PRINT "  GREEN    =G": PRINT
    "  ORANGE   =O": PRINT "  YE
    LLO   =Y": PRINT "  WHILE
     =W": PRINT : PRINT : POKE 3
    4,7
2430 F$ = "FRONT FACE":FA = 4: GOSUB
    2500
2440 F$ = "LEFT FACE":FA = 3: GOSUB
    2500
```

```
2450 F$ = "RIGHT FACE":FA = 5: GOSUB
     2500
2460 F$ = "TOP FACE":FA = 1: GOSUB
     2500
2470 F$ = "BOTTOM FACE":FA = 7: GOSUB
     2500
2480 F$ = "REAR FACE":FA = 10: GOSUB
     2500
2490  TEXT : GOTO 2680
2500  FOR RO = 0 TO 2: FOR CO = 0
      TO 2! PRINT F$", ROW"RO",CO
     LUMN"CO", ";: GOSUB 2510:NC(
     FA,RO,CO) = C: NEXT CO: NEXT
     RO: RETURN
2510  GET C$: IF C$ = "R" THEN C =
     RD
2520  IF C$ = "B" THEN C = BL
2530  IF C$ = "G" THEN C = GN
2540  IF C$ = "O" THEN C = NG
2550  IF C$ = "Y" THEN C = YE
2560  IF C$ = "W" THEN C = WH
2570  IF C$ = "R" OR C$ = "B" OR
     C$ = "G" OR C$ = "O" OR C$ =
     "Y" OR C$ = "W" THEN 2590
2580  PRINT  CHR$ (7);: GOTO 2510

2590  IF C$ = "R" THEN  PRINT "RE
     D"
2600  IF C$ = "B" THEN  PRINT "BL
     UE"
2610  IF C$ = "G" THEN  PRINT "GR
     EEN"
2620  IF C$ = "O" THEN  PRINT "OR
     ANGE"
2630  IF C$ = "Y" THEN  PRINT "YE
     LLOW"
2640  IF C$ = "W" THEN  PRINT "WH
     ILE"
2650  RETURN
2660  HOME : PRINT "DO YOU WANT T
     O CONTINUE PREVIOUS SET UP ?
     ": GET X$: PRINT X$: IF X$ =
     "Y" THEN 2740
2670  REM  INITIALIZE GRAPHICS
2680  GR : POKE  - 16302,0: COLOR=
     BK: FOR Y = 40 TO 47: HLIN 0
```

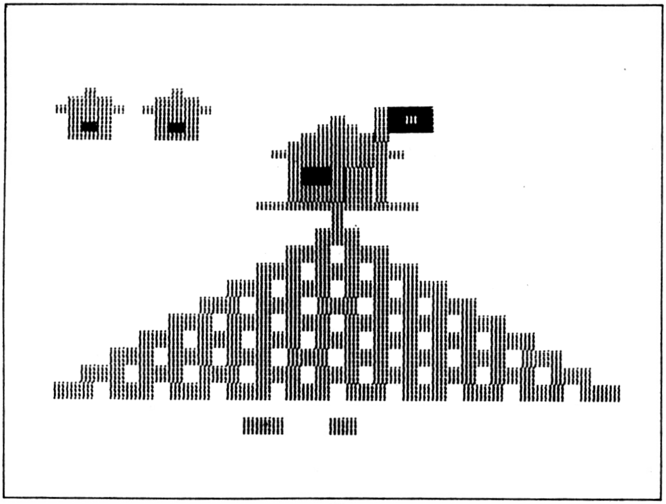
```
     ,39 AT Y: NEXT Y
2690  COLOR= BR
2700  FOR R = 0 TO 2: HLIN 6,31 AT
     R * 12 + 11: NEXT R
2710  FOR C = 0 TO 1: VLIN 0,46 AT
     C * 9 + 14: NEXT C
2720  IF X$ = "Y" THEN 2880
2730  GOTO 2820
2740  REM  READ FROM DISK
2750 D$ =  CHR$ (4): PRINT D$"OPE
     N COLOR CUBE OLD GAME"
2760  PRINT D$"READ COLOR CUBE OL
     D GAME"
2770 F = 3: GOSUB 2800:F = 1: GOSUB
     2800:F = 5: GOSUB 2800:F = 4
     : GOSUB 2800:F = 7: GOSUB 28
     00:F = 10: GOSUB 2800
2780 : PRINT D$"CLOSE COLOR CUBE
     OLD GAME"
2790  GOTO 2680
2800  FOR RO = 0 TO 2: FOR CO = 0
      TO 2: INPUT NC(F,RO,CO): NEXT
     CO: NEXT RO: RETURN
2810  REM  ********************
     DEFAULT INITIALIZATION
2820 FA = 3:CL = RD: GOSUB 2890
2830 FA = 1:CL = BL: GOSUB 2890
2840 FA = 5:CL = YE: GOSUB 2890
2850 FA = 4:CL = GN: GOSUB 2890
2860 FA = 7:CL = NG: GOSUB 2890
2870 FA = 10:CL = WH: GOSUB 2890
2880  GOSUB 340: GOSUB 70: GOSUB
     770
2890  FOR RO = 0 TO 2: FOR CO = 0
      TO 2:NC(FA,RO,CO) = CL: NEXT
     CO: NEXT RO: RETURN
2900 : HOME : PRINT "THIS IS A FL
     AT VERSION OF A THREE
     DIMENSIONAL COLORED CUBE."
2910  PRINT
2920  PRINT "EACH SIDE IS NINE SQ
     UARES AND STARTS    WITH ALL
      SQUARES OF ONE COLOR"
2930  PRINT
2940  PRINT "EACH SIDE MAYBE ROTA
     TED WHICH MIXES    THE COLOR
     S VERY QUICKLY."
```

```
2950  PRINT
2960  PRINT "THE SCREEN SHOWS THE
      FRONT FACE WITH THE LEFT, R
     IGHT, TOP, AND BOTTOM FACES
         SURROUNDING IT AS IF THE
     Y WERE UNFOLDED"
2970  PRINT
2980  PRINT "THE REAR FACE IS SHO
     WN UNFOLDED FROM THE BOTTOM.
     "
2990  PRINT
3000  PRINT "THE LEFT AND RIGHT F
     ACES ARE SHOWN       ROTATED
     ON EITHER SIDE OF THE TOP,
         BOTTOM AND REAR FACES."
3010  VTAB 23: PRINT "HIT SPACE B
     AR FOR MORE": GET Y$: PRINT
     Y$: HOME
3020  PRINT "TO ROTATE A SECTION,
      THE PADDLE INPUTS ARE USED
      IN TWO STEPS (A JOYSTICK IS
        ASSUMED)."
3030  PRINT
3040  PRINT "POSITION THE JOYSTIC
     K IN THE ROW OR      COLUMN O
     F THE FRONT FACE (BLINKING)
         THAT YOU WISH  TO ROTATE
     ."
3050  PRINT "VISUALIZE THE FRONT
     FACE (BLINKING)"
3060  PRINT "WITH THE JOYSTICK IN
      FIVE POSITIONS"
3070  PRINT "LEFT, RIGHT, TOP, BO
     TTOM, AND CENTER"
3080  PRINT
3090  PRINT "                  T"
3100  PRINT "                  ."
3110  PRINT "                  ."
3120  PRINT "              L...C...R"
3130  PRINT "                  ."
3140  PRINT "                  ."
3150  PRINT "                  B"
3160  PRINT : PRINT : PRINT "NOW
     PUSH BOTTOM (0)"
3170  VTAB 23: PRINT "HIT SPACE B
     AR FOR MORE": GET Y$: PRINT
     Y$: HOME
```

```
3180  PRINT "AFTER CHOOSING ONE O
     F THE FIVE THEN"
3190  PRINT "THEN MOVE THE JOYSTI
     CK UP OR DOWN        OR LEFT
     OR RIGHT IN THE DIRECTION OF
         MOVEMENT"
3200  PRINT : PRINT "TO ROTATE TH
     E ENTIRE CUBE USE BOTTOM(1)"

3210  PRINT "WITH THE JOYSTICK IN
      THE CENTER."
3220  PRINT : PRINT "THE MOVE THE
      JOYSTICK IN THE DIRECTION"
3230  PRINT "YOU WISH TO ROTATE"
3240  PRINT : PRINT "AFTER EACH C
     HANGE, THE SPEAKER BEEPS"
3250  PRINT "TO SINGLE IT IS READ
     Y FOR THE NEXT INPUT"
3260  PRINT
3270  PRINT "A SETUP MAY BE INPUT
      MANUALLY AND A      CURRENT
     POSITION MAY BE SAVED TO DIS
     K    AND READ FROM DISK."
3280  PRINT
3290  PRINT " HIT ANY KEY TO STOP
      THE GAME"
3300  VTAB 23: PRINT " HIT SPACE
     TO BEGIN": GET Y$: PRINT Y$
3310  RETURN
```

程式九　金字塔
(PYRAMID)



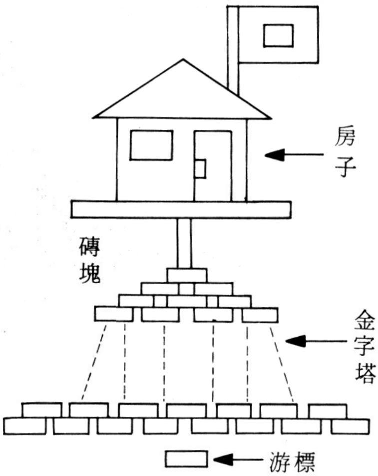
此程式爲最近市面上流行的APPLESOFT BASIC 遊戲程式，有些遊戲程式的磁碟的名稱爲HOUSELESS 。此程式算是難度較深的程式，若沒有詳細的程式說明，便太不容易解讀。因此在本文中對整個程式每一部份做了詳細的分析，若讀者已閱過前面幾章或幾個程式，應可很輕易的配合本文說明解讀此程式。

本程式可算是較典型的COMMERCIAL BASIC GAME，包括程式自動展示玩法（DEMONSTRATION）及正式開始由遊戲者與機器比賽，但也因此而增加了程式的複雜度。由於本程式較

爲複雜，且不太具有結構化程式的要求，很難用流程圖予以整體說明，故本文採用文字敍述配合圖形對程式每一部份做說明，只在較爲複雜的副程式或流程中加上流程圖。

程式中幾個重要觀念說明

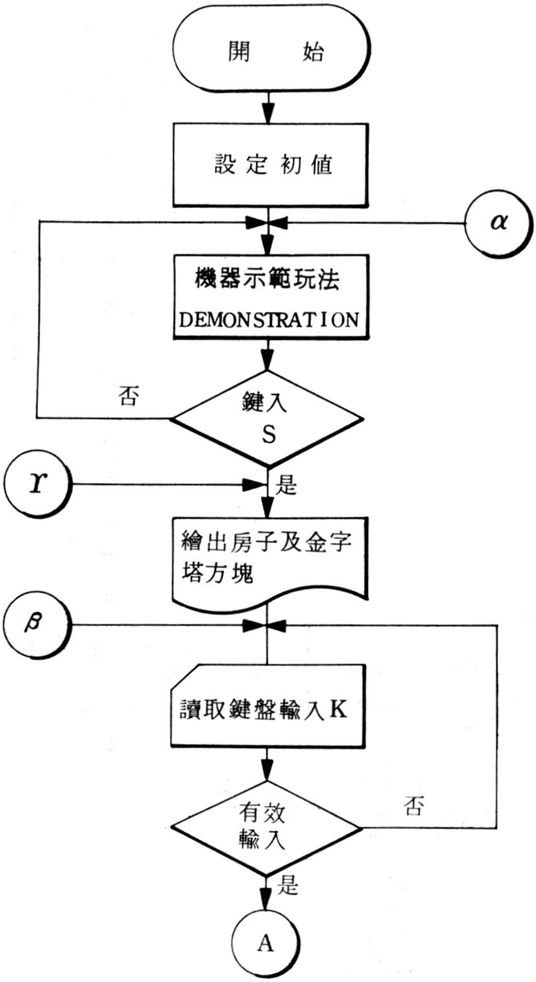
程式一開始是由APPLE 示範本遊戲程式的玩法，首先在螢幕上劃出一座由磚塊堆成的金字塔，塔上有一間房子（如下圖所示）。此金字塔共由五種顏色的磚塊所堆成，每種顏色的磚塊各有不同的重量及得分。APPLE 一次拆除一個磚塊，在拆除過程中可能有些上層的磚塊因此磚塊的移走失去支持力而掉落，甚至使整個金字塔崩潰而使房子倒塌。APPLE 在展示過程中，一塊一塊的將磚塊移走，直到房子與金字塔倒下，便算結束，結束後會從頭再開始示範，直到遊戲者按下“ S ”鍵，便結束示範，正式與遊戲者開始比賽。

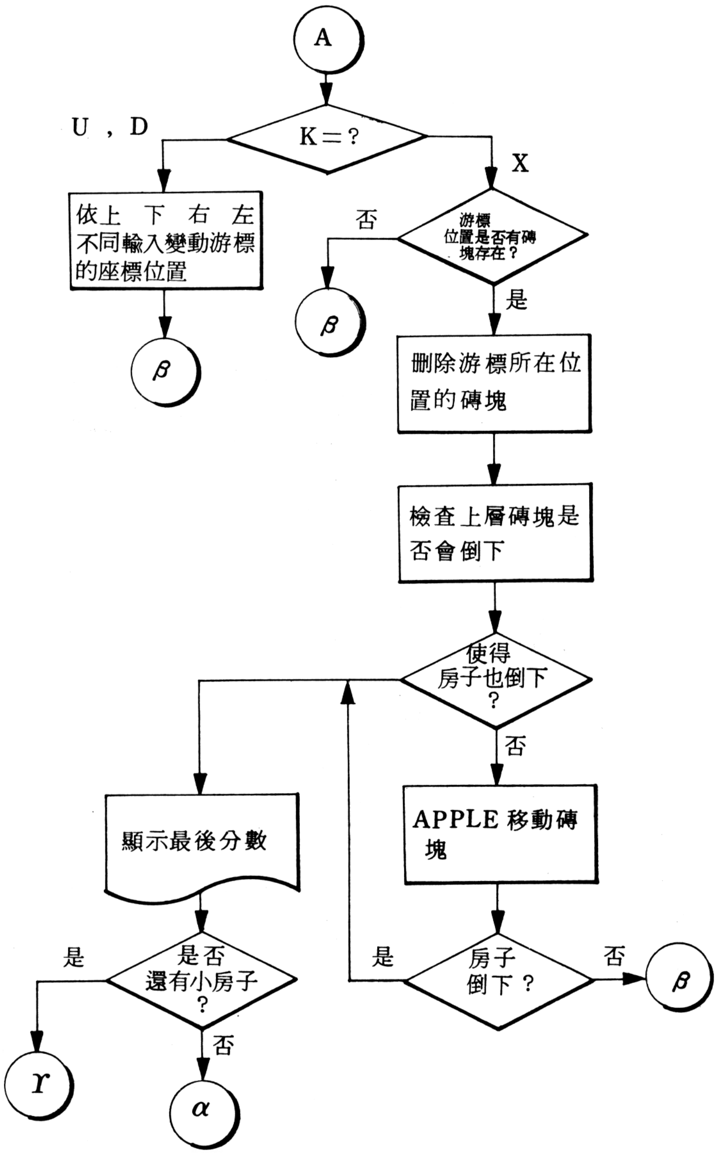


正式開始後遊戲者擁有三棟房子，每次遊戲者弄倒一棟房子後，房子數目便少 1 ，直到三棟房子全部倒塌便算遊戲結束（分數會一直在螢幕下方顯示）。但若房子是由機器弄倒的，則除了再出現一棟房子外，遊戲者所擁有的房子數目並不會減少。遊戲結束後 APPLE 會自動再示範程式的玩法，直到遊戲者再按下“ S ”鍵才又開始比賽。

此外比賽進行中金字塔下方會顯示一閃爍的游標（cursor

），使用者可利用“U”（UP），“D”（DOWN），“→”（RIGHT），“←”（LEFT）使游標移動至所欲拆除的磚塊的位置，再由“X”鍵將此磚塊拆除。整個過程用流程圖表示如下：



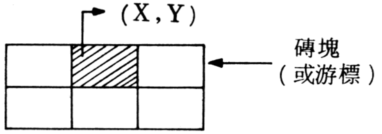

A
U，D
K＝？
X
依上 下 右 左
不同輸入變動游標
的座標位置
β
否
游標
位置是否有磚
塊存在？
β
是
刪除游標所在位
置的磚塊
檢查上層磚塊是
否會倒下
使得
房子也倒下
？
否
顯示最後分數
APPLE 移動磚
塊
是
是否
還有小房子
？
否
r
α
是
房子
倒下？
否
β

流程一：整個程式的流程

僅由遊戲中，我們無法看出此遊戲程式撰寫的基本想法（因此可以說，許多遊戲者事實上並不了解此遊戲的玩法，不過這也許是撰寫者的目的：使遊戲者無法獲得高分）。以下便是此程式設計時的想法：

(1)金字塔磚塊的表示法及游標的移動：

螢幕上的游標與磚塊相同均由六個低解像圖形的點構成（只是顏色不同），而以中央上層的座標爲整個磚塊或游標的座標如右圖：因此游標每移動一個位置其座標的變量爲 2；



其中 $0 \leq X \leq 36$，$16 \leq y \leq 34$ 爲磚塊之座標範圍

“ U ”：UP游標上移，若 $y > 17$，則 $y \leftarrow y - 2$。

“ D ”：DOWN 游標下移，若 $y \leq 36$，則 $y \leftarrow y + 2$。

“→”：RIGHT 游標右移，若 $x \leq 35$，則 $x \leftarrow x + 2$。

“←”：LEFT 游標左移，若 $X \geq 3$，則 $x \leftarrow x - 2$。

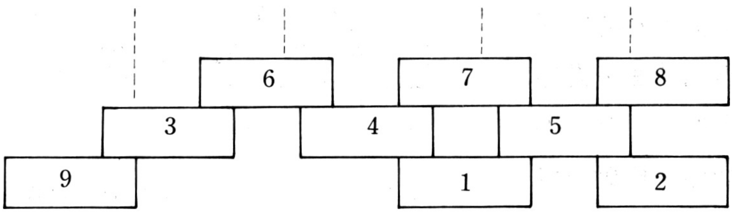
在改變座標（X，Y）之前須先檢查（X，Y）之值是否已到達邊界，是否能夠移動。

磚塊的大小及座標均使用上述方法表示，但事實上共有五種磚塊，這五種磚塊主要是用不同的顏色供遊戲者區分。這五種磚塊各有不同的重量及不同的得分，因此遊戲者的得分是按照其所移走及崩落的磚塊種類之不同而獲得不同的分數，在程式中是以陣列變數 PR 儲存每一種磚塊所代表的分數。此外各種磚塊的重量亦以陣列 WA 儲存供程式中判斷。

(2)磚塊掉落與否的判斷：

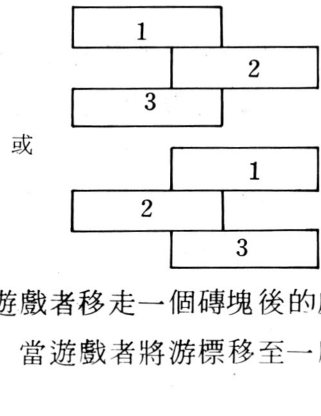
往往移掉一個磚塊會導致上層磚塊的崩落，甚至一層接一層崩落使整棟房子的倒塌，但到底移走一塊磚塊會不會引起上層磚塊的崩落，或是崩落到什麼程度呢？這是整個程式最重要的觀念。

有些情形遊戲者可以很容易的判斷出來，但有些情況便須額外的資料幫助我們判斷。如下圖中，將磚塊 1 移走必定會引起磚



塊 4 的崩落，至於磚塊 5 是否崩落，則須視磚塊 8 的重量是否壓得住磚塊 5；且磚塊 4 落下後磚塊 6 可能也會崩落，須視磚塊 6 上一層的情形而定，但磚塊 3、9、2 則必定不會崩落。

判斷一個磚塊會不會因支持不住的原則必須考慮二個因素，如下圖說明：



在左邊的情況中要保持磚塊 2 不掉下必須：（WA 代表重量）

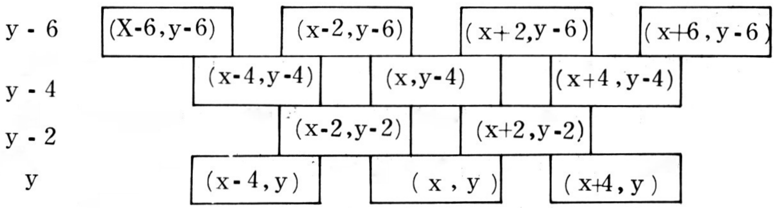
① WA (1)＞ WA (2)

及②磚塊 3 必須存在；才能支持磚塊 2。

(3)遊戲者移走一個磚塊後的處理步驟：

當遊戲者將游標移至一座標為（ X ，Y ）的磚塊，並按下“

X ”鍵後，程式除了將此磚塊移走還要考慮一個問題：那些上層的磚塊也會因支持不住而掉下？房子是否會倒塌？第二個問題很容易回答，只要最上層的磚塊也落下了，房子便倒塌了。但第一個問題應如何解決呢？程式的處理方式如下：



在上圖中對（X，Y）被遊戲者移出後，程式將（X－2，Y－2）及（X＋2，Y＋2）二個磚塊座標存入陣列S（1－2）中，代表還有二個磚塊必須要檢查；接著將S（1－2）陣列之各數值，改存至陣列SS(1—2)中，並將S陣列清除。此後對SS陣列的二個磚塊逐一檢查，如先檢查（X－2，Y－2）磚塊會不會落下，若會便將（X－4，Y－4）與（X，Y－4）兩磚塊存入S（1－2）中，並將（X－2，Y－2）磚塊移走，接著再檢查（X＋2，Y－2）磚塊會不會落下，若會便再將（X，Y－4）與（X＋4，Y－4）存入陣列S（3－4）中，至此SS 陣列檢查完畢。再重覆將S（1－4）存入SS(1—4)中，並將S清除，再逐一對SS 的各數值檢查，並再重新在S陣列中存入更上一層須檢查的磚塊座標。如此反覆，直到S陣列中已沒有新的數值或已到達金字塔的頂端（崩潰），便可將所有可能受（X，Y）磚塊移走而落下的各磚塊找出並清除掉。

由上述討論，我們便可知道陣列S的作用為儲存目前的移走的磚塊的上一層所有必須檢查是否會落下的磚塊；而SS 陣列則

用以儲存目前被移走磚塊同一層所有必須被檢查的磚塊。而將S陣列內容存入 SS 陣列中便是一層檢查完後再向上一層檢查。此外陣列S與 SS 均僅儲存磚塊的X座標，因同一層磚塊的Y座標均相同，向上移一層時僅須將原來磚塊的Y座標值減 2 便可適用於所有同一層的磚塊。

(4)房子倒塌的處理：

房子倒塌時，先是房子慢慢的往下降，然後房子向左邊倒下，並由 TEXT MODE 與 GRAPHIC MODE 交替在螢幕上顯示，所以會出現一閃一閃的情況，此外配合呼叫發聲副程式，發出房子崩潰的聲音，最後顯示最後的得分，並檢查遊戲者是否還有房子，若三間房子均已倒塌，程式便回到示範玩法等待遊戲者按下“ S ”鍵，重新開始。

此外還有一點需要說明即在拆除磚塊的過程中，若掉落的磚塊愈多，所得的分數便愈高，但很可能導致房屋的倒塌，因此遊戲者每次在移動一磚塊時，均需經過一番深思熟慮。讀者研究過此程式後將可發現，此程式的玩法並不像表面上那麼容易，事實上極具趣味性。

重要變數說明

1. CO(0－15):COLOR ，陣列內所存的是 5 種顏色值。其初值分別爲 2，13，13，10，10，10，8，8，8，4，4，4，4，4 （ 2 －深藍色， 4 －深綠色， 8 －棕色，13－黃色，10－灰 2 色），此陣列主要作用是在程式中由RND內存程式隨意組成不同的金字塔結構（即不同磚塊的分佈）。
2. PR(0－15):POINT RECEIVE，爲一陣列用以儲存五種顏色不同的磚塊所代表的不同分數。 PR (S)表示顏色碼爲S的

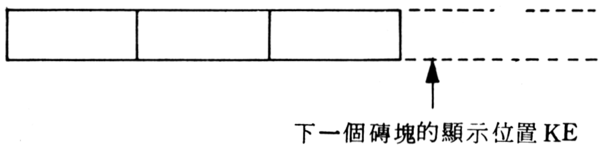
磚塊所含的分數，往後此種顏色的磚塊被移走或掉下時，便可得到 PR (S)的分數。事實上 PR 當常數使用且僅用到 PR (2)＝5，PR (4)＝0，PR (8)＝1，PR (13)＝3，PR (10)＝2 爲了方便才宣告PR(0－15）。與 CO 陣列的初值比較我們知道，出現機率較少的磚塊其分數愈高。

3. WA(0－15)：WEIGHT，作用與 PR 陣列差不多，只是WA 所代表的是各種不同磚塊的重量，亦當常數使用其初值爲 WA (2)＝2，WA (4)＝1，WA (8)＝3，WA (10)＝4，WA (13)＝5。

4. S(0－10)，SS(0－10)： 此二陣列在程式中均沒有宣告，APPLESOFT自動將其DIM設定爲 S (10)及 SS (10)。SS 與 S 陣列的作用，已在前面說明不再重覆。

5. P，PS：P，PS 分別爲陣列 S 與 SS 的指標，指出陣列 S 與SS 的元素個數（即有幾個磚塊須進行檢查）。

6. KE：程式中對每一塊落下或被移走的磚塊均在螢幕底端顯示，而 KE 的作用便是指示所移走或掉落的磚塊在螢幕底端的顯示位置。



7. PO：POINT 記錄總分。

8. GP：記錄一次移走磚塊所獲得分數，故每次移動磚塊後均須將 GP 加入 PO 中·

9. DE：DEMDNSTRATION FLAG，當旗標用。DE=1時代表程式目前在示範玩法·DE＝0時代表示已正式開始。

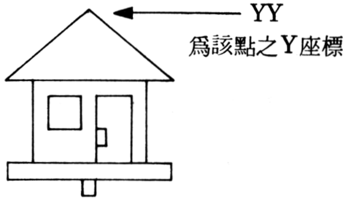
10. IE：記錄遊戲者尚擁有的房子數目。初值爲 3，每次遊戲者弄

倒一棟房子後，IE 便減1。由於此程式是由日本人所撰寫有些變數名稱及程式中之註解REM均使用日文唸法拼成英文如屋子日文發音類似〔i:e〕故將表示房子的變數名稱拼成IE。

11. T：因在程式與遊戲者比賽時，須判別輪到遊戲者或APPLE移動磚塊，故使用此變數記錄：T＝1時代表APPLE移動磚塊，T＝0時代表輪到遊戲者移動磚塊。

12. YY：決定房子在螢幕上的Y座標值，如圖：

初值 YY＝0，當房子往下掉時 YY 值增加。

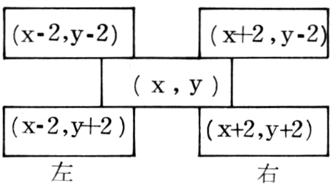


13. DO：DOWN，記錄當一磚塊被移走後其上層磚塊會不會落下。DO＝1時表示會，DO＝0時表示不會。

14. (X,Y),(IX,IY),(TX,TY)：均指示方塊的位置座標。

程式流程說明

(1) 900－930行：決定座標爲(X,Y)的磚塊能否被支持住不落下，DO 變數用以儲存檢查的結果 DO＝0不會落下，若(X,Y)磚塊會落下則將 DO 設定爲1。其檢查方式爲：(X,Y)爲被檢查之磚塊。



程式中使用SCRN函數決定(X,Y)位置是否存在磚塊，及其顏色爲何？SCRN(X,Y)＝0代表(X,Y)位置無磚塊存在。

步驟：

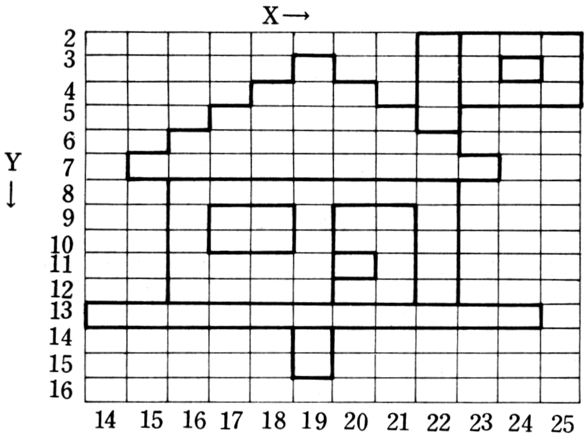
(i)若 SCRN(X,Y)=0，則 DO=0 return 。

(ii)左邊：SCRN(X−2,Y+2)≠0 且 WA(SCRN(X−2,Y−2)>WA(SCRN(X,Y)) 則 DO=0 ，return 。

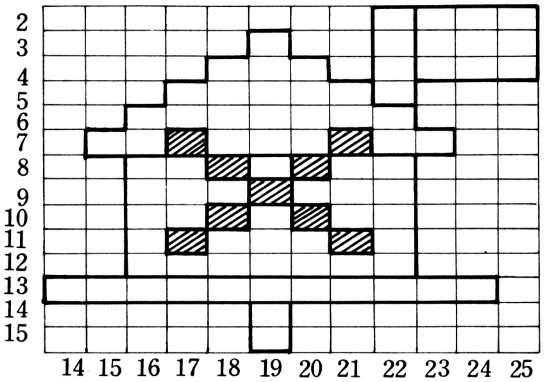
(iii)右邊：SCRN(X+2,Y+2)≠0 且 WA(SCRN(X+2,Y−2)>WA(SCRN(X,Y)) 則 DO=0 ，return 。

(iv)其他情況均表示（X ，Y）無法被支持住 DO ＝ 1 。

(2) 1000 − 1090 行：在螢幕上繪出金字塔頂端的房子。此程式的房子有二種，一種是輪到遊戲者移動磚塊時螢幕上所顯示的房子；另一種是機器移動時螢幕上所顯示的房子。在本副程式中所繪的是前者：（其順序為由上向下進行）。



(3) 1100 − 1190 行：在螢幕上繪出金字塔頂端的房子。此副程式所繪出者為 APPLE 移動磚塊時螢幕上所顯示的房子。如下圖。其繪圖方式亦是由上向下繪出。



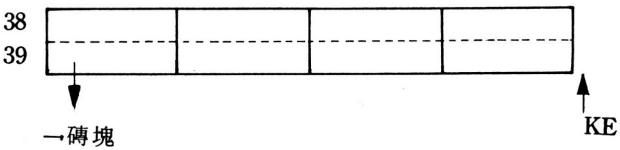
(4) 1200－1240 行：將座標爲（X，Y）之磚塊由螢幕上清除，並計算其得分，GP＝GP＋PR(S)，S爲（X，Y）磚塊的顏色（但若此磚塊是由APPLE所移走便不計算其分數）。分數每次更動後會在螢幕上顯示，主要步驟如下：

(i)計算分數及顯示得分。

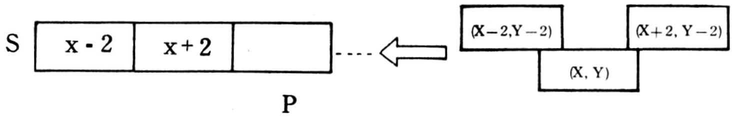
(ii)發出聲音（POKE 781,18：CALL 768）。

(iii)將座標爲（X，Y）之磚塊清除。

(iv)將移走的磚塊在螢幕底端（螢幕垂直座標38，39）顯示：以KE爲指標，當KE＞36即已塡滿一行後就不再顯示，否則KE＝KE＋3。

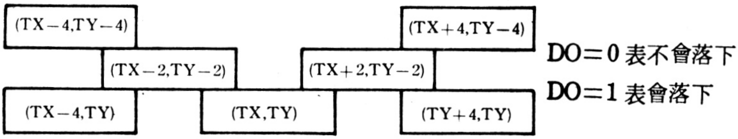


(5) 1300－1320行：將（X，Y）座標的磚塊上一層兩個方塊的X座標存入S陣列中，指標P指示存的位置S(P)。



(6) 1400 — 1420行：將陣列S中所有元素移至陣列SS中。

(7) 1500 — 1570行：決定座標（X，Y）的磚塊被移走後金字塔上其他磚塊是否會有崩下的可能。此副程式是APPLE 在決定移動那一個磚塊時的考慮方式。



測試（TX，TY）移走時其他磚塊會不會落下的步驟爲：

(i) SCRN(TX-2,TY-2)=0 且 SCRN(TX+2,TY-2)=0 則DO＝0 。
此種情況代表（TX，TY）爲獨立磚塊。

(ii) SCRN(TX-2,TY-2)≠0 且 SCRN(TX-4,TY)≠0 且 SCRN(TX-4,TY-4)>SCRN(TX-2,TY-2) 則 DO=0；左邊支持得住。

(iii) SCRN(TX+2,TY-2)≠0 且 SCRN(TX+4,TY+4)≠0 且 SCRN(TX+4,TY-4)>SCRN(TX+2,TY-2) 則 DO＝0；右邊支持得住。

(iv) 其他情況，（TX，TY）磚塊的移走會引起其他磚塊的落下，故不能移走。

(8) 1600 — 1680 行：爲顯示磚塊將被移走前的訊號。其方式是

將要移走的磚塊以橘色閃爍：

(i)將要移走的磚塊，顯示成橘色。

(ii)發聲（POKE 781,17：CALL 768）。

(iii)顯示磚塊原來的顏色。

(iv)發聲。如此重覆步驟(i)－(iv)四次。

(9) 1700 － 1799 行：左螢幕左上角顯示遊戲者剩餘的房子。在 APPLE 示範玩法時（DEMONSTRATION）會跳過此副程式。

(10) 10000－ 11000 行：設定初值，WA，PR，CO 三陣列，並將發聲機器語言存入記憶體第三頁區。

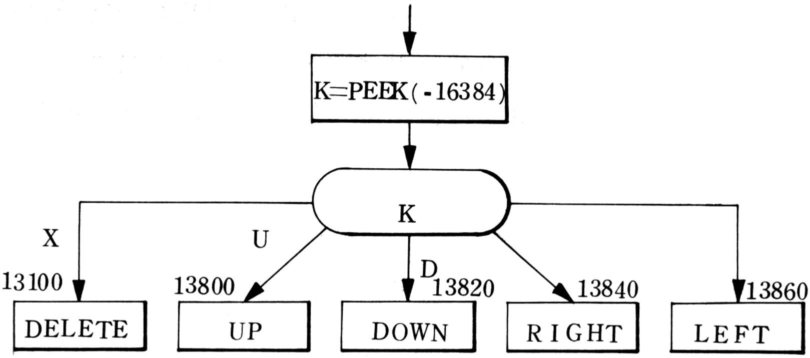
(11) 11900－ 11910 行：爲正式開始遊戲的進入點。設定遊戲開始的初值 PO ＝ 0（總分爲 0）；DE ＝ 0（正式比賽）；IE ＝ 3（有三間房子）。

(12) 12000－ 12008行：爲示範玩法（DEMONSTRATION）時的進入點，否則以 11900 行爲進入點。印出示範時在螢幕底端的訊息。

(13) 12010 － 12060 行：顯示金字塔－－共由十層磚塊所堆成。每次產生的金字塔均不相同。各磚塊的顏色由：

CO(INT(RND(14)*14)) 產生 0 至13的任意數值後，對應至 CO 陣列找出不同的顏色，因此每次才能有不同的起始狀況。

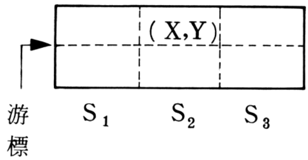
(14) 13020 － 13050 行：根據鍵盤輸入移動游標。



⒂ 13100 — 13200 行：檢查欲移走磚塊的位置上是否有磚塊存在，若無則回到 13020 行接受新的輸入，否則將 PO 減 1，GOTO 至 14000 行將該磚塊清除。

⒃ 13800 — 13870 行：依照鍵盤輸入改變游標之座標值。

⒄ 13900 — 13950 行：用以顯示游標目前所在的位置。其方式爲：⑴將構成游標的下面 3 個低解像點的顏色分別存入變數S_1，S_2，S_3中。

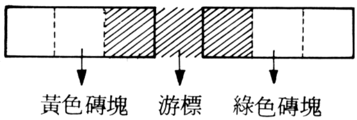


⑵將游標以橘色顯示並暫停一小段時間。

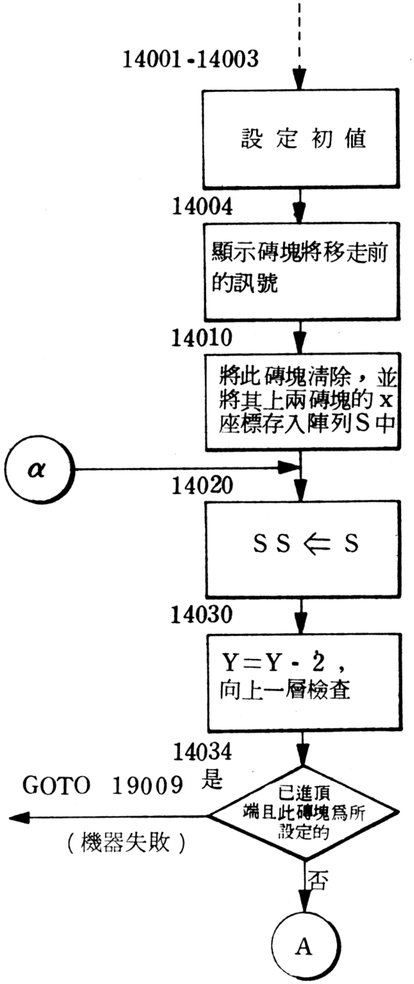
⑶恢復原來的原色。

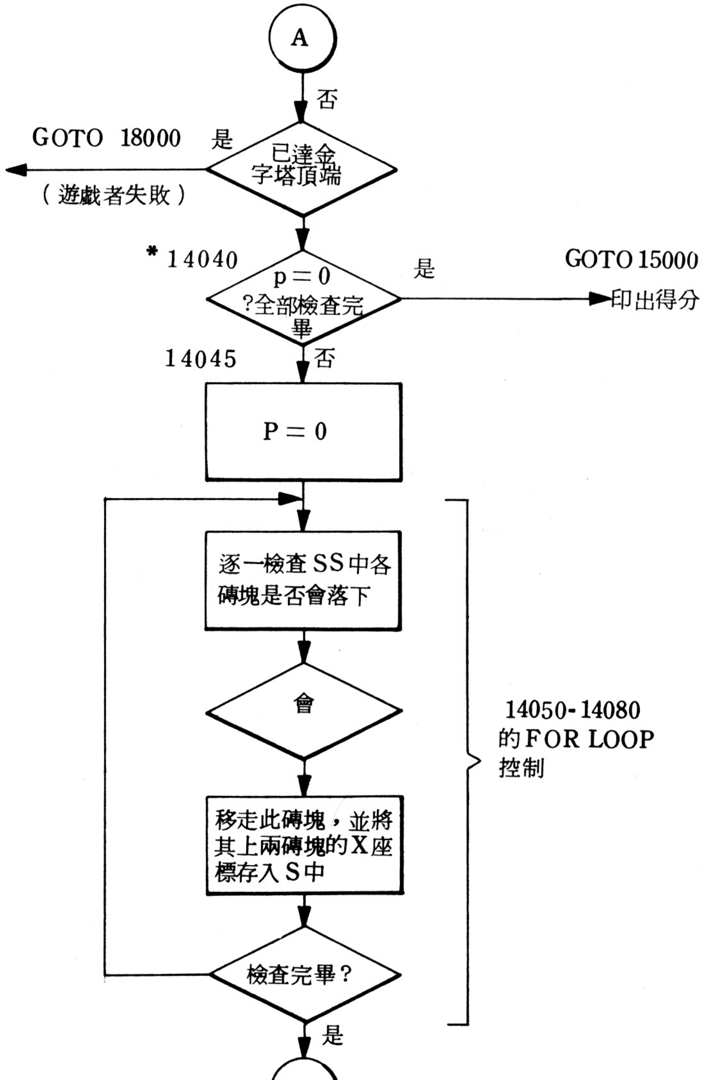
所以要將各顏色存入S_1，S_2，S_3中的原因爲：可能游標的顯示位置正好横跨在不同顏色的兩磚塊中央。如下圖：

此時當游標消失後，必須能將磚塊根據其原來的顏色繪出。



(18) 14000－14090行：執行移去磚塊之動作，同時必須將受影響而會落下的磚塊一起移走。其步驟如下（X，Y為“X”鍵被按下時，游標所在之位置座標，即欲移走的磚塊之座標）：




A
否
GOTO 18000
是
已達金字塔頂端
（遊戲者失敗）
* 14040
p＝0
?全部檢查完畢
是
GOTO 15000
印出得分
14045
否
P＝0
逐一檢查SS中各磚塊是否會落下
會
移走此磚塊，並將其上兩磚塊的X座標存入S中
檢查完畢？
是
α
14050-14080
的FOR LOOP
控制

在上述流程中變數P用作陣列S的指標，故當P＝0時代表S中已無須檢查的磚塊（上一層無可能落下的磚塊），故程式中利用P＝0判別是否檢查完畢。

(19) 15000 － 15060 行：計算此次移動所得的分數。並有"BONUS"，即GP ≥ 20 → GP ＝ GP*4
GP ≥ 15 → GP ＝ GP*3
GP ≥ 10 → GP ＝ GP*2
（以上三式）最後將PO← PO+GP

此外在印分數時亦會發聲。

(20) 15900 － 15940 行：在螢幕上顯示GP與PO 兩分數。

(i) GP ＝ 0　不顯示，否則以正常方式（NORMAL）顯示"＊GPOO ＊"，GP 爲其GP 所存之分數。

(ii) PO ＜ 10，以反白方式（INVERSE）顯示
"0 0 PQ .O"
PO ＜ 100，以反白方式顯示"0 PQ0 0"
PO ≥ 100，以反白方式顯示" PQ0 0"
其中PQ 爲PQ 中所存之分數。上面的方式主要用以調整數字的長短，使顯示出相同的位數。

(21) 16000 － 17010 行：APPLE 移動磚塊。16004 行設定房屋的頂端在（X，Y）＝（19，3），並畫出房子。在程式展示玩法時雖爲APPLE 單獨移動磚塊，但顯示的是遊戲者移動時所顯示的房子。（TX，TY）爲游標之座標值。

APPLE 找尋將移走的磚塊的方式爲：由金字塔最下一層的最右一塊磚頭向左並向上一層層尋找仍存在的磚塊，並由1500 行的副程式判斷移走此磚塊後上層磚塊是否會塌下。若會則不移動此磚塊，繼續重複向左及向上找尋新的磚塊，否則便移走此磚塊（GOSUB 1600及 GOSUB 1200）。

接著程式判斷此時是屬於示範（DEMONSTRATION）或正式開始，若為正式比賽則將螢幕底端所顯示的所有摔落磚塊清除，再回到 13000 行接受遊戲者的動作；否則程式暫停一小段時間，察看遊戲者是否已按下“S”鍵結束示範，若在此暫停期間“S”鍵沒有被按下，則至 13000 行後再跳回此段程式，繼續未完的示範。可用流程圖將程式從開始示範至示範結束的流程以流程圖表示在下一頁，但此流程僅說明示範（DEMON）之過程，而不是整個程式的流程。

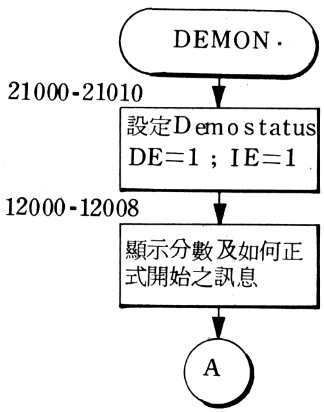
(22) 18000 － 18200行：在螢幕上顯示遊戲者的房子倒塌的效果。其中 18010 － 18020行將房子清除；18030 行在新位置畫出房子，並發聲。且呼叫下面幾個副程式進行房屋倒塌時的背景效果。

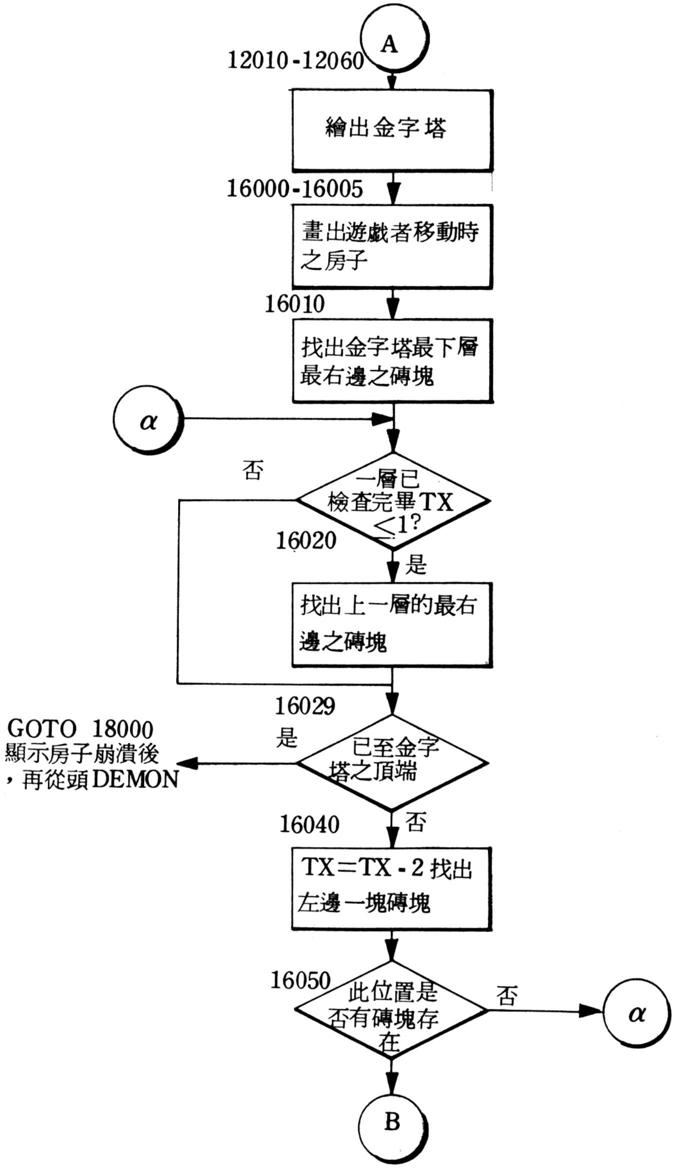
18890 － 18891 行：繪出倒塌之屋頂。

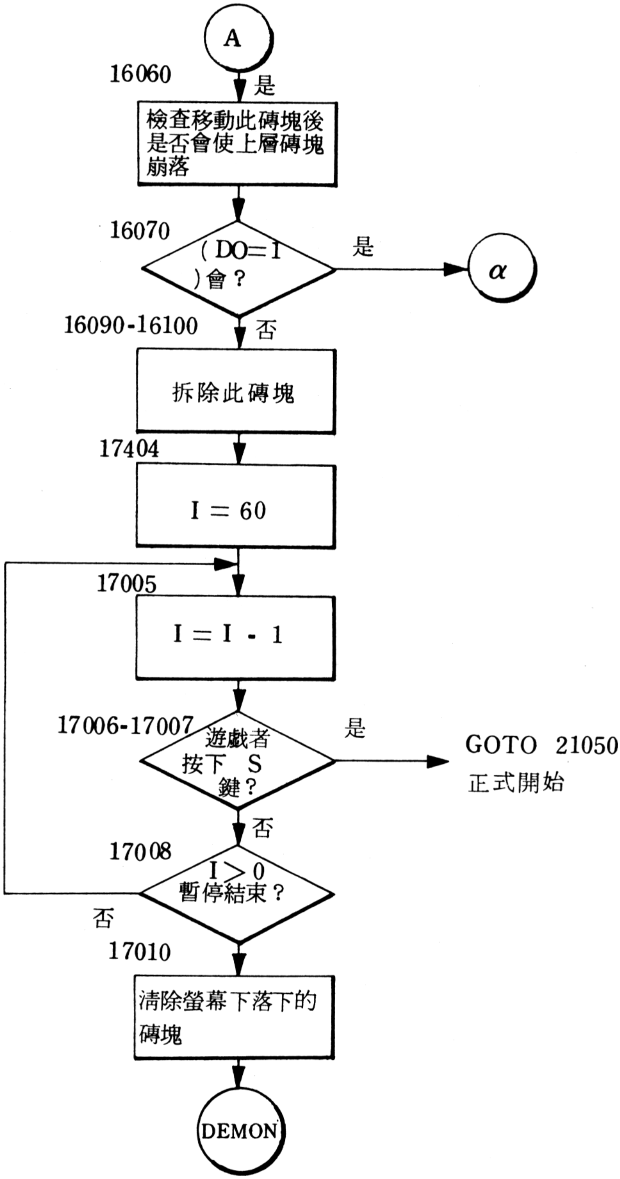
18893 － 18894 行：繪出旗子的旗面。

18896 － 18896 行：繪出房子的牆壁。

18897 及 18899行均為製造房屋倒塌之效果（見下）。




A
12010-12060
繪出金字塔
16000-16005
畫出遊戲者移動時之房子
16010
找出金字塔最下層最右邊之磚塊
α
否
一層已檢查完畢TX≤1?
16020
是
找出上一層的最右邊之磚塊
16029
GOTO 18000
顯示房子崩潰後，再從頭DEMON
是
已至金字塔之頂端
16040
否
TX=TX-2 找出左邊一塊磚塊
16050
此位置是否有磚塊存在
否
α
B


A
16060
是
檢查移動此磚塊後是否會使上層磚塊崩落
16070
（DO=1）會？
是
α
16090-16100
否
拆除此磚塊
17404
I = 60
17005
I = I - 1
17006-17007
遊戲者按下 S 鍵？
是
GOTO 21050
正式開始
否
17008
I > 0 暫停結束？
否
17010
清除螢幕下落下的磚塊
DEMON

(23) 18950 – 18980 行：製造房屋倒塌的背景效果。

將螢幕以 TEXT及 GR 交替顯示，並加入音響效果，以產生天崩地裂的感覺。其中 POKE – 16303 ，0 ，設定 TEXT ；POKE – 16304 ，0 ，設定GR 。

(24) 19000 – 19200 行：APPLE 弄倒房子後的房子倒塌情形（類似上面的過程）。

(25) 20000 – 20120 行：爲房子倒塌後的進入點。IE = IE – 1 決定遊戲者是否還有房子可繼續進行遊戲。否則印出：

<<< FINAL SCORE —— PQOO >>> 暫停

暫停後由程式進行示範（DEMON ）。

(26) 21000 – 21010 行：設定 DEMON 之初值DE = 1 ；IE = 1 並至12000 行進行示範。

(27) 21040 行的 GOTO 21010 ，21010 爲 APPLE 進行DEMON 之進入點。21050 行的 GOTO 11900 ，11900 爲APPLE 與遊戲者開始進行比賽的進入點。

最後還有一點需要說明，在程式中 10020 行的POKE 214，128 ：的作用主要是對程式進行保護，使遊戲者利用RESET 或 CTR - C 將遊戲停止後，無論鍵入任何APPLESOFT 指令（不包括磁碟作業系統的指令），均變成RUN而使程式再自動執行，當然也 LIST不出來。因此想要LIST 程式必須在程式開始執行前鍵入 LIST。

```
]LIST

1  GOTO 10000
900 DO = 0: IF  SCRN( X,Y) = 0 THEN
     RETURN
910  IF  SCRN( X - 2,Y + 2) <  >
    0 AND WA( SCRN( X - 2,Y - 2)
    ) > WA( SCRN( X,Y)) THEN  RETURN

920  IF  SCRN( X + 2,Y + 2) <  >
    0 AND WA( SCRN( X + 2,Y - 2)
    ) > WA( SCRN( X,Y)) THEN  RETURN

930 DO = 1: RETURN
1000  REM <IE>
1010  COLOR= 8: FOR I = 0 TO 4: HLIN
    19 - I,19 + I AT 3 + I + YY:
     NEXT I
1020  COLOR= 4: FOR I = 8 TO 12: HLIN
    16,22 AT I + YY: NEXT I
1030  COLOR= 3: HLIN 17,18 AT 9 +
    YY: HLIN 17,18 AT 10 + YY
1035  COLOR= 14: VLIN 9 + YY,12 +
    YY AT 20: VLIN 9 + YY,12 + Y
    Y AT 21
1040  COLOR= 2: PLOT 20,11 + YY
1045  COLOR= 6: HLIN 14,24 AT 13 +
    YY: VLIN 14 + YY,15 + YY AT
    19: VLIN 2 + YY,5 + YY AT 22

1050  COLOR= 15: FOR I = 2 TO 4: HLIN
    23,25 AT I + YY: NEXT I
1055  COLOR= 9: PLOT 24,3 + YY
1090  RETURN
1100  REM <TEKI IE>
1110  COLOR= 5: FOR I = 0 TO 4: HLIN
    19 - I,19 + I AT 3 + I + YY:
     NEXT I
1120  COLOR= 5: FOR I = 8 TO 12: HLIN
    16,22 AT I + YY: NEXT I
1145  COLOR= 6: HLIN 14,24 AT 13 +
    YY: VLIN 14 + YY,15 + YY AT
    19: VLIN 2 + YY,5 + YY AT 22
1150  COLOR= 5: FOR I = 2 TO 4: HLIN
    23,25 AT I + YY: NEXT I
```

```
1160  COLOR= 0: PLOT 17,7 + YY: PLOT
     21,7 + YY: PLOT 18,8 + YY
1165  PLOT 20,8 + YY: PLOT 19,9 +
     YY: PLOT 18,10 + YY: PLOT 20
     ,10 + YY: PLOT 17,11 + YY: PLOT
     21,11 + YY
1190  RETURN
1200  REM <<KESI>>
1203 S =  SCRN( X,Y)
1204  IF T = 1 THEN 1208
1205 GP = GP + PR(S)
1207  GOSUB 15900
1208  COLOR= 0
1209  POKE 781,18: CALL 768
1210  FOR I = 0 TO 1: HLIN X - 1,
     X + 1 AT I + Y: NEXT I
1215  GOSUB 5000
1220  IF KE > 36 THEN 1240
1230 KE = KE + 3: COLOR= S: HLIN
     KE - 2,KE AT 38: HLIN KE - 2
     ,KE AT 39
1240  RETURN
1300  IF X - 2 >  = 0 THEN S(P) =
     X - 2:P = P + 1
1310  IF X + 2 <  = 39 THEN S(P) =
     X + 2:P = P + 1
1320  RETURN
1400  FOR L2 = 0 TO P - 1
1410 SS(L2) = S(L2): NEXT L2
1420 PS = P: RETURN
1500 DO = 0
1505  IF TX < 4 THEN 1550
1510  IF  SCRN( TX - 2,TY - 2) =
     0 THEN 1550
1515  IF  SCRN( TX - 4,TY) <  > 0
      AND  SCRN( TX - 4,TY - 4) >
      SCRN( TX - 2,TY - 2) THEN 1
     550
1520 DO = 1: RETURN
1550  IF  SCRN( TX + 2,TY - 2) =
     0 THEN  RETURN
1560  IF  SCRN( TX + 4,TY) <  > 0
      AND  SCRN( TX + 4,TY - 4) >
      SCRN( TX + 2,TY - 2) THEN  RETURN
1570 DO = 1: RETURN
```

```
1600  REM  ** TORU MAE **
1620 S =  SCRN( X,Y)
1630  FOR J = 1 TO 4
1640  COLOR= 9: HLIN X - 1,X + 1 AT
     Y: HLIN X - 1,X + 1 AT Y + 1

1649  POKE 781,17
1650  CALL 768
1660  COLOR= S: HLIN X - 1,X + 1 AT
     Y: HLIN X - 1,X + 1 AT Y + 1

1670  CALL 768
1675  NEXT J
1680  RETURN
1700  REM  << IE KAZU >>
1704  COLOR= 0
1705  FOR I = 0 TO 5: HLIN 0,16 AT
     I: NEXT I
1710  IF IE = 1 THEN  RETURN
1720  FOR J = 1 TO IE - 1
1730 X = (J - 1) * 6: GOSUB 1790
1740  NEXT J
1790  COLOR= 8: PLOT X + 2,0: HLIN
     X + 1,X + 3 AT 1: HLIN X,X +
     4 AT 2
1792  COLOR= 4: FOR I = 3 TO 5: HLIN
     X + 1,X + 3 AT I: NEXT I
1794  COLOR= 7: PLOT X + 2,4
1799  RETURN
5000  REM  ** KESI **
5010  RETURN
10000  REM  << INIT >>
10010  DIM CO(15),PR(15),WA(15)
10020  POKE 214,128
10100  FOR I = 0 TO 13: READ CO(I
     ): NEXT I
10102  DATA   2,13,13,10,10,10,8,
     8          ,8,4,4,4 ,4,4
10105 WA(2) = 2:WA(13) = 5:WA(10)
      = 4:WA(8) = 3:WA(4) = 1:WA(
     0) = 0
10110 PR(2) = 5:PR(13) = 3:PR(10)
      = 2:PR(8) = 1:PR(4) = 0
10120  FOR I = 0 TO 46: READ J: POKE
     768 + I,J: NEXT I
```

```
10130  DATA   169,1,133,16,169,0,
     133,17,169,140,133,18,164,17
     ,173,48,192,152,170,202,208,
     253
10132  DATA  200,196,18,208,243,1
     6              5,18,173,48,192
     ,152,170,202,              208
     ,253,136,196,17,208,243,1
               98,16,208,222,96
11000  GOTO 21000
11900 PO = 0
11905 DE = 0
11910 IE = 3:TI = 0
12000  REM  << INIT YAMA >>
12005 : GR : HOME
12006  GOSUB 15900
12008  IF DE = 1 THEN  HOME : FLASH
     : PRINT " NOW DEMONSTRATION.
     ";: NORMAL : PRINT " TYPE 'S
     ' FOR START": PRINT : INVERSE
     : PRINT " COPYRIGHT 1980 STA
     R CRAFT INC.          ";: NORMAL

12010  FOR Y = 1 TO 10
12020 X = (10 - Y) * 2
12030  FOR I = 1 TO Y: COLOR= CO(
      INT ( RND (14) * 14))
12040  HLIN X,X + 2 AT Y * 2 + 14

12050  HLIN X,X + 2 AT Y * 2 + 15
     :X = X + 4: NEXT I
12060  NEXT Y
12070  GOSUB 1700
13000  REM  << INPUT >>
13003  IF DE = 1 THEN 16000
13005 YY = 0: GOSUB 1000
13010 IX = 19:IY = 38
13020  GOSUB 13900
13030 K =  PEEK ( - 16384): IF K <
     128 THEN 13020
13035  POKE  - 16368,0
13040  IF K = 213 THEN 13800
13042  IF K = 196 THEN 13820
13044  IF K = 136 THEN 13840
13046  IF K = 149 THEN 13860
```

```
13048  IF K = 216 THEN 13100
13050  GOTO 13020
13100  IF  SCRN( IX,IY) = 0 THEN
     13020
13110 X = IX:Y = IY
13120  GOSUB 13700
13130 PO = PO - 1: IF PO < 0 THEN
     PO = 0
13140  GOSUB 15900
13200  GOTO 14000
13700  REM  ** INPUT **
13710  RETURN
13800  IF IY > 17 THEN IY = IY -
     2
13810  GOTO 13020
13820  IF IY <  = 36 THEN IY = IY
      + 2
13830  GOTO 13020
13840  IF IX >  = 3 THEN IX = IX -
     2
13850  GOTO 13020
13860  IF IX <  = 35 THEN IX = IX
      + 2
13870  GOTO 13020
13900 S1 =  SCRN( IX - 1,IY):S2 =
      SCRN( IX,IY):S3 =  SCRN( IX
      + 1,IY)
13910  COLOR= 9: HLIN IX - 1,IX +
     1 AT IY: HLIN IX - 1,IX + 1 AT
     IY + 1
13920  FOR I = 1 TO 30: NEXT I
13930  COLOR= S1: VLIN IY,IY + 1 AT
     IX - 1
13932  COLOR= S2: VLIN IY,IY + 1 AT
     IX
13933  COLOR= S3: VLIN IY,IY + 1 AT
     IX + 1
13950  RETURN
14000  REM  << KESI >>
14001 KE = 0
14002 T = 0
14003 GP = 0
14004  GOSUB 1600
14005 P = 0
14010  GOSUB 1200: GOSUB 1300:
```

```
14020  GOSUB 1400
14030 Y = Y - 2
14034  IF T = 1 AND Y = 14 THEN 1
     9009
14035  IF Y = 14 THEN 18000
14040  IF P = 0 THEN 15000
14045 P = 0
14050  FOR L1 = 0 TO PS - 1
14060 X = SS(L1): GOSUB 900
14070  IF DO = 1 THEN  GOSUB 1200
     : GOSUB 1300
14080  NEXT L1
14090  GOTO 14020
15000  REM  << POINT >>
15015  IF GP = 0 THEN 15100
15020  IF GP >  = 20 THEN GP = GP
      * 4: GOTO 15030
15022  IF GP >  = 15 THEN GP = GP
      * 3: GOTO 15030
15024  IF GP >  = 10 THEN GP = GP
      * 2: GOTO 15030
15026  IF GP < 10 THEN 15040
15030  VTAB 21: HTAB 10: FLASH : PRINT
     "!!! BONUS !!!"
15032  FOR I = 1 TO 3000: NEXT I
15035  GOSUB 15900
15036  FOR I = 1 TO 3000: NEXT I
15040  IF GP > 9 THEN GP = GP - 1
     0:PO = PO + 10: GOTO 15045
15041 GP = GP - 1:PO = PO + 1
15045  GOSUB 15900
15050  CALL 768
15060  IF GP > 0 THEN 15040
15100  COLOR= 0
15110  FOR I = 38 TO 39: HLIN 0,3
     9 AT I: NEXT I
15800  GOTO 16000
15900  REM  POINT
15910  HOME : IF GP = 0 THEN  PRINT
     : GOTO 15920
15913  PRINT "*";GP;"00*"
15920  PRINT
15925  INVERSE : PRINT " ";
15926  IF PO < 10 THEN  PRINT "0"
     ;
```

```
15927  IF PO < 100 THEN  PRINT "O
     ";
15930  PRINT PO;"OO "
15935  NORMAL
15940  RETURN
16000  REM  << TEKI KESI >>
16001 T = 1
16003 YY = 0
16004 X = 19:Y = 3
16005  IF DE = 1 THEN  GOSUB 1000
     : GOTO 16010
16006  GOSUB 1100
16010 TX = 39:TY = 34
16020  IF TX <  = 1 THEN TY = TY -
     2:TX = 37
16029  IF DE = 1 AND TY <  = 18 THEN
     18000
16030  IF TY <  = 18 THEN 19000
16040 TX = TX - 2
16050  IF  SCRN( TX,TY) = 0 THEN
     16020
16060  GOSUB 1500
16070  IF DO = 1 THEN 16020
16080 X = TX:Y = TY
16090  GOSUB 1600
16100  GOSUB 1200
17000  REM
17004 I = 60
17005 I = I - 1
17006 K =  PEEK ( - 16384)
17007  IF DE = 1 AND K =  ASC ("S
     ") + 128 THEN 21050
17008  IF I > 0 THEN 17005
17010  COLOR= 0: FOR I = 38 TO 39
     : HLIN 0,39 AT I: NEXT I
17100  GOTO 13000
18000  REM  << IE KUZUSHI >>
18001 Y = 0
18010  COLOR= 0: FOR I = YY + 2 TO
     YY + 16
18020  HLIN 14,25 AT I: NEXT I
18030 YY = YY + Y:Y = Y + 1: GOSUB
     1000
18040  GOSUB 18950
18050  IF YY < 21 THEN 18010
```

```
18060  GOSUB 18900
18070  GOSUB 18899:C = 8:X = 6:Y =
      15: GOSUB 18890:C = 15:X = 2
      5:Y = 7: GOSUB 18893:
18075  COLOR= 9: PLOT 27,9
18080 C = 4:X = 20:Y = 23: GOSUB
      18896
18085  COLOR= 7: HLIN X + 2,X + 3
       AT Y + 1: HLIN X + 2,X + 3 AT
      Y + 2: HLIN X + 1,X + 3 AT Y
       + 4: HLIN X + 1,X + 3 AT Y +
      5
18090  GOSUB 18900
18200  GOTO 20000
18800  REM  ** IE TOBI **
18890  REM  YANE
18891  COLOR= C: FOR I = 0 TO 4: VLIN
      Y - I,Y + I AT X + I: NEXT I
      : RETURN
18893  REM  HATA
18894  COLOR= C: FOR I = 0 TO 3: HLIN
      X + 1,X + 3 AT Y + I: NEXT I
      : PLOT X,Y: RETURN
18896  REM  IE
18897  COLOR= C: FOR I = 0 TO 4: VLIN
      Y,Y + 6 AT X + I: NEXT I: RETURN

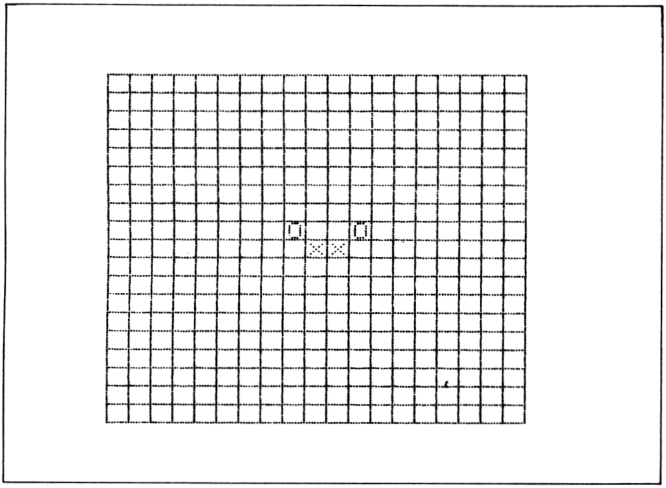
18899  COLOR= 0: FOR I = 23 TO 39
      : HLIN 15,25 AT I: NEXT I: RETURN

18900  FOR I = 1 TO 3: GOSUB 1895
      0: NEXT I: RETURN
18950  REM
18955  POKE  - 16303,0
18960  CALL 768
18970  POKE  - 16304,0
18975  CALL 768:
18980  RETURN
19000  REM << TEKI KUZUSHI >>
19002 X = 19:Y = 20: GOTO 14004
19009 Y = 0
19010  COLOR= 0: FOR I = YY + 2 TO
      YY + 16
19020  HLIN 14,25 AT I: NEXT I
19030 YY = YY + Y:Y = Y + 1: GOSUB
```

```
      1100
19040  FOR I = 1 TO 10:K =  PEEK
      ( - 16336): NEXT I
19050  IF YY < 21 THEN 19010
19053  GOSUB 18900
19055  GOSUB 18899:C = 5:X = 27:Y
       = 10: GOSUB 18890:X = 8:Y =
      10: GOSUB 18893:X = 12:Y = 2
      1: GOSUB 18896
19060  GOSUB 18900
19199  IF DE = 1 THEN 21020
19200  GOTO 12000
20000  IF DE = 1 THEN 21000
20001 IE = IE - 1
20010  IF IE <  = 0 THEN 20100
20020  GOTO 12000
20100  HOME : PRINT "<<< FINAL SC
      ORE -- ";PO;"00 >>>"
20110  FOR I = 1 TO 5000: NEXT I
20120  GOTO 21000
21000  REM  DEMO
21010 DE = 1:IE = 1: GOTO 12000
21020  REM
21040  GOTO 21010
21050  GOTO 11900

]FP
```

程式十　五子棋 (GOBANG)



五子棋是我國固有的一種遊戲，它可以算是一種大型的井字棋，使用圍棋盤的一部份（19×19）來玩。當遊戲者任何一方產生五個或五個以上相連，不論直、橫或是斜的就算獲勝。

本文所討論的五子棋程式，雖然棋力不是十分強，但遊戲者若不特別小心還是會失敗。事實上，作者提出這份程式討論的目的，是希望藉此程式告訴讀者棋類程式應如何撰寫，並希望讀者在了解程式的結構後，能加以修改，使本程式更具攻擊力（本程式注重在防守）。

程式介紹

本程式利用高解像圖形、文字混合幕顯示圖形和訊息與遊戲者溝通，至於程式的內部運作主要是使用一個優先等級表和一些特殊狀況來決定機器應將棋子下在棋盤上的那一個位置。程式中主要根據前二次棋子所下的位置（一次爲機器所下，另一次爲遊戲者所下），檢查其八個方向上的前後四個位置。對這些位置計算其棋子下了之後的優先等級（若新的優先等級大於原有的優先等級，則以新的等級取代原有的優先等級），最後再檢查所有的位置選出優先等級最高的位置，做爲機器下棋子的位置。

由上面的說明，讀者可能已經注意到整個程式的主要關鍵便在優先等級的決定，而此優先等級主要是依據程式優等級表和一些特殊情況的判斷（說明見後），因此若此優先等級表及少數特殊狀況能考慮周詳便能提高程式的棋力。故作者極力推薦讀者試著去修改此程式，以創造出一個更具攻擊力的“五子棋”程式。

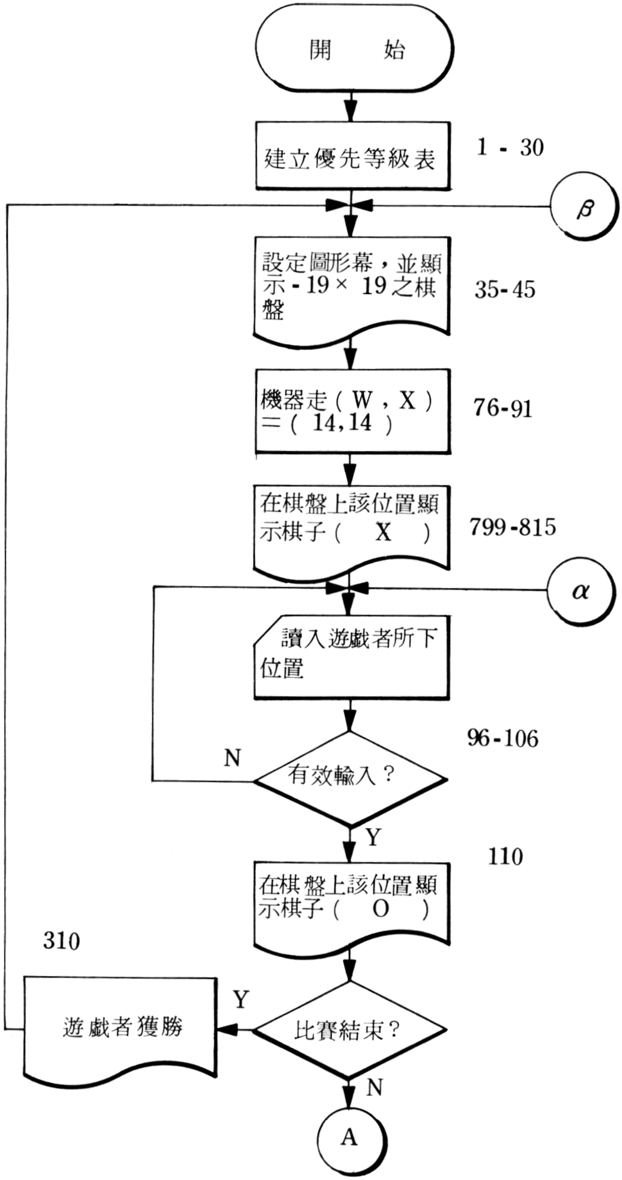
程式的主要流程

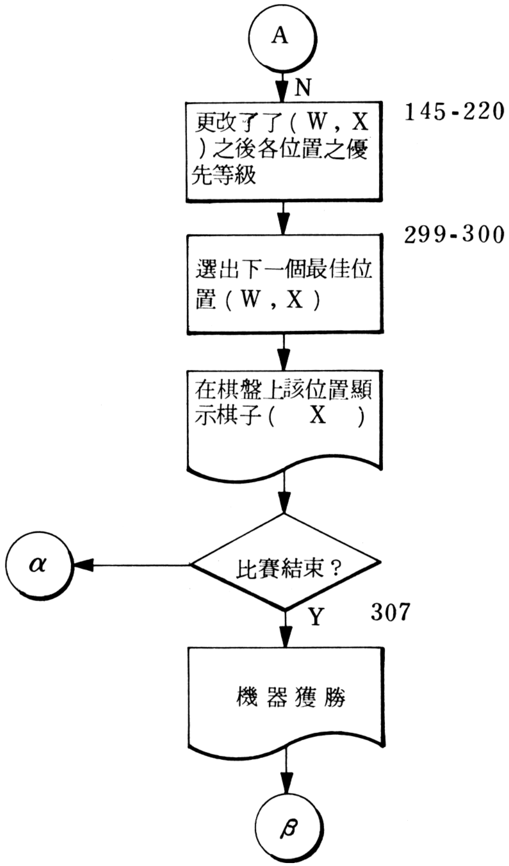
本程式的流程很簡單，可用流程圖表示如下頁。遊戲進行後首先由機器下，機器所下的棋子爲“X”，遊戲者爲“O”，輪流進行直到分出勝負後，遊戲便重新開始。

程式中主要變數

(1) T（27,27）：用來代表棋盤，記錄棋盤上進行的狀態。T陣列中存放的值主要有三種（0，1，2）。當此陣列宣告後，APPLESOFT 會將其中所有元素全部設定爲0，代表空白棋盤。若機器在棋盤中某位置下了一棋子（“X”），則

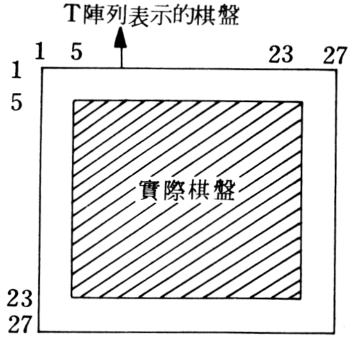
程式流程圖：


開　始
建立優先等級表
1 - 30
β
設定圖形幕，並顯示 - 19× 19 之棋盤
35-45
機器走（W，X）=（14,14）
76-91
在棋盤上該位置顯示棋子（X）
799-815
α
讀入遊戲者所下位置
96-106
N
有效輸入？
Y
110
在棋盤上該位置顯示棋子（O）
310
Y
遊戲者獲勝
比賽結束？
N
A



T陣列中的對應位置便被設定爲1；若爲遊戲者所下（“O”）則T陣列中的對應位置便被設定爲2，用以區別。實際上的棋盤僅有19×19，多出來的八行與八列主要是因爲在計算優先等級時，必須由某一特定位置向八個方向伸展四個位

置。雖然在接近棋盤邊緣位置不一定能向各方向伸展，但爲了避免區別那些位置可伸展幾個位置，故決定將棋盤擴大（如圖右）。事實上右圖中空白位置在眞正尋找機器下一次所下的位置時並不會被搜尋到。



因此若要檢查在螢幕棋盤上遊戲者所輸入的位置座標（X，Y）是否爲空白時，應T（X＋4，Y＋4）＝0（注意X，Y座標的變換）。

(2)M（19，19）：儲存棋盤（19×19）的每個位置上的優先等級值。其起始值由APPLESOFT全部設定爲0，往後每當機器或遊戲者下了一個棋子之後，M陣列就要根據其所下的位置更換一次。當任何一個位置之優先等級值爲100時（M（I，J）＝100），表示比賽已結束，有五個同類棋子已排成一列。

(3)S（81）：爲程式中所使用的優先等級表，根據棋面上棋子排列的型式，而有不同的優先等級值。機器決定所下的棋子位置主要即依此表而來（見次頁）

當對棋子的幾種排列方式建立了優先等級之後，還須能很快的從優先等級表中找出某種排列情況的優先等值級是多少。爲了達到這個目的，除了將優先等級表排列如前之外，在程式中並使用一特殊的數學模式來決定棋子各種排列方式的優先等級。即利用三進位計數，並將空白位置用0代表，“X”用1代表，“O”用2代表，而將某一種棋子排列方

優先等級表：S（81）

指標	類型	優先等級值	指標	類型	優先等級值	指標	類型	優先等級值
0	↑————	0	27	↑×———	15	54	↑○———	0
1	↑———×	0	28	↑×——×	20	55	↑○——×	0
2	↑———○	0	29	↑×——○	10	56	↑○——○	0
3	↑——×—	0	30	↑×—×—	40	57	↑○—×—	0
4	↑——××	0	31	↑×—××	50	58	↑○—××	0
5	↑——×○	0	32	↑×—×○	30	59	↑○—×○	0
6	↑——○—	0	33	↑×—○—	0	60	↑○—○—	21
7	↑——○×	0	34	↑×—○×	0	61	↑○—○×	11
8	↑——○○	0	35	↑×—○○	0	62	↑○—○○	41
9	↑—×——	0	36	↑××——	39	63	↑○×——	0
10	↑—×—×	40	37	↑××—×	65	64	↑○×—×	0
11	↑—×—○	0	38	↑××—○	40	65	↑○×—○	0
12	↑—××—	30	39	↑×××—	70	66	↑○××—	0
13	↑—×××	47	40	↑××××	100	67	↑○×××	0
14	↑—××○	0	41	↑×××○	60	68	↑○××○	0
15	↑—×○—	0	42	↑××○—	30	69	↑○×○—	0
16	↑—×○×	0	43	↑××○×	30	70	↑○×○×	0
17	↑—×○○	0	44	↑××○○	30	71	↑○×○○	0
18	↑—○——	0	45	↑×○——	0	72	↑○○——	31
19	↑—○—×	0	46	↑×○—×	0	73	↑○○—×	11
20	↑—○—○	1	47	↑×○—○	0	74	↑○○—○	41
21	↑—○×—	0	48	↑×○×—	0	75	↑○○×—	0
22	↑—○××	0	49	↑×○××	0	76	↑○○××	0
23	↑—○×○	0	50	↑×○×○	0	77	↑○○×○	0
24	↑—○○—	1	51	↑×○○—	0	78	↑○○○—	51
25	↑—○○×	0	52	↑×○○×	0	79	↑○○○×	40
26	↑—○○○	21	53	↑×○○○	0	80	↑○○○○	90

※ ↑下此位置優先等級值；“ × ”機器所下的子；“ ○ ”遊戲者所下之子；“ — ”表空白

式看成一三進位數字，再將其換算成十進位數值便可至優先等級表的對應項去查得其優先等級值。如：

$$\uparrow \text{OOXX} \rightarrow 2\times3^3+2\times3^2+1\times3^1+1\times3^0=76$$

$$3^3 3^2 3^1 3^0$$

再由優先等級表的S（76）＝0便可知箭頭位置的優先等級值爲0。又如：

$$\uparrow \text{XXOO} \rightarrow 1\times3^3+1\times3^2+2\times3^1+2\times3^0=44$$

$$3^3 3^2 3^1 3^0$$

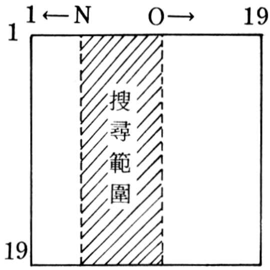
S（44）＝30，故箭頭位置的優先等級值即爲30。

(4) C：用來記錄程式進行狀況的變數，其初值爲－1，當遊戲者將五個棋子連成一線後，其值便改爲0，表示遊戲者獲得勝利。

(5)（W，X）：爲座標變數，用來表示機器所下棋子的位置座標。此最佳位置是由M（19，19）中所找出優先等級值最高的位置。

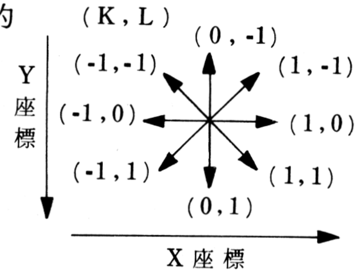
(6)（Y，Z）：爲遊戲者所下的位置座標。

(7) N，O：機器在M（19，19）陣列中搜尋最佳的下子位置時，爲了避免在沒有影響的位置搜尋，影響程式的執行速度，故將搜尋範圍訂個上、下限N即爲其下限；O爲搜尋的上限。在程式中N，O受棋子所下位置所改變，我們留待後面程式中再討論。



(8) K，L：當程式中對某一位置決定其優先等級值時，必須對八個方向進行檢查，而此K，L即用來當做旋轉時的X，Y座

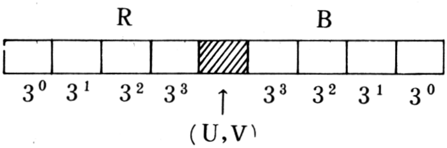
標的變量，故其值共有右圖中的幾種情形。



(9) D：為一計數器，其值為 0 至 5，主要用來控制程式某一中心位置，向八個方向各向前進行四個位置求出其優先等級值。

(10) A：用來暫存某特定位置的優先等級值。

(11) B，R：因在對某一位置決定其優先等級時，必須由該位置向其前後四個位置決定其排列方式（如右圖，圖中陰影位置即為優先等級表中的箭頭位置。故



R＝T（U－K，V－L）＊27＋T（U－2＊K，V－2＊L）＊9＋T（U－3＊K，V－3＊L）＊3＋T（U－4＊K，V－4＊L）

B＝T（U＋K，V＋L）＊27＋T（U＋2＊K，V＋2＊L）＊9＋T（U＋3＊K，V＋3＊L）＊3＋T（U＋4＊K，V＋4＊L）

(12) S：為程式中 800 行的副程式的參數。在呼叫 800 行的副程式之前須先將 S 存入適當數值，使 800 行的繪圖副程式能依情況繪出"O"或"X"（S＝1 代表"X"；S＝2 代表"O"）。

(13)（PI，PJ）：為棋盤上欲繪出"O"或"X"的位置座標，其值均 1 ≤ PI，PJ ≤ 19，亦為 800 行的繪圖副程式之參數。

⒁（SI，SJ）：為 800 行的繪圖副程式所使用的變數，其值為棋盤上（PI，PJ）位置所對應的眞正高解像螢幕的座標位置 $0 \leq SI \leq 279$；$0 \leq SJ \leq 191$。

程式流程說明

在後面的程式中為五子棋程式的BASIC 程式部份，還有一個圖形表（SHAPE TABLE）存在記憶體的第三頁區：

```
*300: 02 00 06 00 17 00 08 0C
      0C DF DB 0E 0E 0E 0E 0E
      0E DF DB 0C 0C 04 00 1B
      24 0C 2D 15 36 36 1E 3F
      07 20 04 00
```

此圖形表包括二個圖形，圖形一為“X”（7 × 7 點組成），圖形二為“O”（5 × 7 點組成）。在正式執行此程式之前必須存入記憶體第三頁區中。

流程說明：

(1) 1 －30行：設定優先等級表。在程式第一行設定LOMEN：16384，使變數存入高解像圖面第二頁記憶區之上，因程式中需使用第一頁高解像螢幕記憶區。雖然程式本身並不長，但因使用到許多陣列，若不將變數儲存區往上移勢必會使部份變數被存入高解像螢幕第一頁的記憶區，而破壞圖形，或使得原來變數的內容被後來所畫的圖形所破壞。程式中11至30行即為前面所說明過的優先等級表。

(2)35－45行：設定繪圖的參數，並在高解像螢幕上顯示一19×19的棋盤。棋盤上每個格子由 7 × 7 個點組成，故整個棋盤的繪圖範圍為：$60 \leq X \leq 212$，$0 \leq Y \leq 152$。

(3)97－120行：接受遊戲者所下的棋盤座標。包括下列幾個步驟：

(i)讀入遊戲者所下的座標（Y，Z）。

(ii)檢查此座標位置是否在棋盤內，若不是則重新輸入。

(iii)檢查所下的位置是否已有棋子，若有則重新讀入。

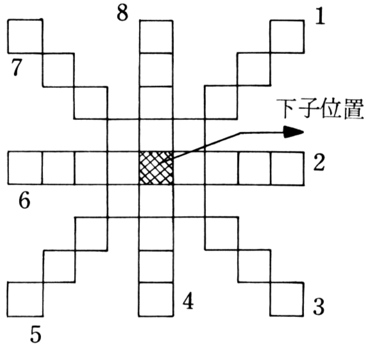
(iv)在T陣列內設定此位置為2，代表“O”。並呼叫繪圖副程式在螢幕上的棋盤顯示出“O”

(4) 145－220行：在棋盤上選出最好（優先等級最高）的位置做為機器所下的位置（W，X）。為了避免浪費時間搜尋一些不可能的位置，故將搜尋範圍限制在上、下限間（N－1與O＋1之間）。200行檢查選出的位置是否為空白。210比較新位置與原先所選出位置的優先等級。

(5) 299－310行：299－300行顯示機器所下的位置座標並在螢幕上棋盤中顯示出來。301－310行檢查比賽是否已結束(i)若M（W－4，X－4）＝100時代表機器已將五個棋子連成一串，獲得勝利，並重新開始新的一個回合(ii)若C＝0即M（Y－4，Z－4）=80表示遊戲者獲勝，重新開始新的一個回合。

(6) 1000－1130行：此副程式是整個程式的核心部份，用來改變棋盤上相關空白位置的優先等級值以供程式中145至220行選擇出最佳部份。不論遊戲者或機器下了一個新子之後，此子周圍的空白位置的優先等級值就將受影響而必須予以改變，若新的優先等級值大於原有的優先等級值，則以新的優先等級值取代。其更改優先等級值的方式如下：以遊戲者或機器所下的位置為中心點向八個方向，每個方向向前延伸四個位置，若這些位置為

空白子則比較其優先等級值或予以更改（如右圖所示）。



右圖中所示總共有32個位置須求其優先等級，圖中的數字代表各方向的求取次序。首先沿方向 1 由靠近中心點的位置向上分別求取，接著換方向 2………。程式中 1089 至 1130 行即爲當一方向全部求取完畢後改變到下一個方向，其中以K，L指示方向（見前面變數K，L之說明），1090 行代表方向 8 已求取完畢跳出此副程式。

變數D用以控制某一方向上由靠近中心點的位置逐漸向上延伸四個求取位置。而對每一個位置（圖中的每一空白方格）決定其優先等級之過程如下：

(i)求得該位置之B與R（見前面B、R之變數說明）。

(ii)根據特殊狀況表，檢查目前該位置是否處於一種特殊型式中，若是則更改其優先等級值爲特殊狀況時的優先等級值（程式中更改R即爲更改優先等級值，因S(R)即爲優先等級值）。

(iii)若求得之值較原有的值高則更改成較高之優先等級值。

步驟(i)中求得的B與R主要用來了解該位置的左、右棋子的排列型式，以便在步驟(ii)中檢查是否屬於一特殊型式。程式 1042 至 1068 行即爲檢查特殊型式，後頁中我們將其對應的特殊型式列出幫助讀者了解（見下頁）。此外我們應注意到程式中 1070 行是以S(R)當做該位置的優先等級值

，且若該位置符合某一特殊情況時更改的是R值，均沒有使用到B值，這是因爲在計算R、B值時，R爲靠近中心點（即靠近下子位置）的四個棋子的型式，影響最大，而B值則爲另一端之四個子，與遊戲者或機器的下子位置較沒有關連。

最後還有一點須要說明的是上、下限O、N之決定。在程式中 1002及 1003 行根據每一次下子位置的X座標來調整上、下限。其代表的意義如下：

(i)程式一開始，首先設定上、下限爲14，14（棋盤中央）。

(ii)當遊戲者所下的位置之X座標超出此上、下限時，便將此上、下限向上或向下移動。

設定此上、下限在遊戲剛開始時可以節省不少時間，因機器在棋盤中央下了一子之後，遊戲者的下了位置必定也在棋盤中央不遠，如此便限定此上、下限在靠近棋盤中央，可節省程式中 145 － 220 行不少搜尋時間，但隨著遊戲的進行，使上、下限逐漸向上或向下移動。由於上、下限間爲所有棋子集中的部份，界限之外沒有棋子存在，故程式不去考慮該區域的優先等級。

特殊型式表：

程式行數	型　式	優先等級	表S中存放 相同等級之位置
1042 行	×↑－××	65	37
1044 行	－0↑00	90	80
1046 行	00↑00	90	80

1048行	0 ↑ 0 —	31	72
1050行	0 ↑ 0 — 0 —	50	31
1053行	0 ↑ 0 0 0 —	90	80
1054行	0 ↑ 0 0 0 ×	90	80
1056行	0 ↑ × × × 0	0	81
1058行	× ↑ × × —	60	41
1058行	× ↑ × × ×	60	41
1059行	× ↑ × — ×	60	41
1060行	0 0 ↑ 0 —	90	80
1061行	× ↑ × × ×	100	40
1062行	× × ↑ × ×	100	40
1063行	× × × ↑ ×	100	40
1064行	— 0 0 0 —	41	

有一端已下了“×”，即成爲× 0 0 0　或 0 0 0 ×，故將優先等級降成41

1067行　有兩列“0”相交的位置，可以會形成“雙三”的型式，故將該位置的優先等級提高爲41

1068行　有兩列“×”相交的位置，很可能會形成“雙三”的型式，故提高該位置的優先等級以避免。

↑：下此位置（即檢查的位置）　0：遊戲者所下子

×：機器所下子　—：表示未定情況，已下“0”，“×”或未下

結語

五子棋是一種很吸引人的遊戲（只是本程式在效果上還不夠好，似乎應加上一點“音效”增加趣味性），若要得勝就必須用心去玩，仔細思考每一步所下的子。一個具有強力攻擊力的程式

，其處理過程也相對的較爲複雜，所要考慮的情況也愈多。本程式中並沒有使用到什麼特殊技巧，整個程式的核心部份即在決定最佳下子位置，像這類程式就是所謂的「人工智慧」（ Artifical Intelligence ）的一個例子。程式設計的重點偏重於使機器具有判斷能力，根據棋盤上的情況，使用最少的時間，選出一個最佳的攻擊（防守）位置。而要使程式思考時間減短，便須使用很簡單的判斷規則，本程式所用的優先等級表就是一個方法，只可惜此表無法包括所有的情況，且有些例外情形，故必須加上特殊情況處理，當然特殊情況愈多將使程式的執行速度減低。但以本程式而言增加特殊情況的處理將使程式的攻擊力增加，雖然在功能與時效上我們必須做一個選擇，但以一般人工智慧程式而言功能重於時間，增加一些時間加強程式的能力還是值得的。若讀者有興趣加強此五子棋程式的能力，可以試著增加特殊情況的處理，創造出一個新五子棋程式。

最後，還有一點需要指出，即程式中沒有檢查是否整個棋盤均已填滿旗子的情況，因在一19×19的棋盤上不太可能發生這種情形。

```
]LIST

1  LOMEM: 16384: REM  GOBANG
2  REM  M IS ARRAY HOLDING BEST M
     OVE
3  REM  T IS BOARD, S IS PRIORITY
      OF THAT POSITION
4  DIM M(19,19),T(27,27),S(81)
5  REM  SET UP PRIORITIES
7  FOR I = 1 TO 81
8 S(I) = 0
9  NEXT I
```

```
11 S(20) = 1:S(10) = 40:S(12) = 3
    0
14 S(13) = 47:S(27) = 15:S(28) =
    20
16 S(29) = 10:S(30) = 40:S(31) =
    50
18 S(32) = 30:S(24) = 1:S(36) = 3
    9
20 S(37) = 65:S(38) = 40:S(39) =
    70
22 S(40) = 100:S(41) = 60:S(42) =
    30
24 S(43) = 30:S(44) = 30:S(62) =
    41
26 S(72) = 31:S(73) = 11:S(74) =
    41
28 S(78) = 51:S(80) = 90:S(26) =
    21
30 S(79) = 40:S(60) = 21:S(61) =
    11
35  HGR : HCOLOR= 3: SCALE= 1: ROT=
    1: CALL  - 936
37  POKE 232,0: POKE 233,3
40  FOR I = 60 TO 212 STEP 8
41  HPLOT I,0 TO I,152
42  NEXT I
43  FOR J = 0 TO 152 STEP 8
44  HPLOT 60,J TO 212,J
45  NEXT J
76 C =  - 1:W = 14:N = 14
87 O = 14:X = 14
88  REM  MAKE FIRST MOVE
91  GOTO 299
96  REM  REQUEST MOVE AND CHECK F
    OR VALIDITY
97  VTAB 22: HTAB 5: INPUT "YOUR
    MOVE X= ";Y: VTAB 23: HTAB 1
    5: INPUT "Y= ";Z
99 Y = Y + 4:Z = Z + 4
101  IF Y > 23 THEN 97
102  IF Z > 23 THEN 97
103  IF Y < 5 THEN 97
104  IF Z < 5 THEN 97
106  IF T(Y,Z) > 0 THEN 97
110 T(Y,Z) = 2:PI = Y - 4:PJ = Z -
    4:S = 2: GOSUB 800
```

```
115 I = Y
120 J = Z
125  REM  STUDY LAST TWO MOVES
127  GOSUB 1000
128  IF C <  >  - 1 THEN  GOTO 31
     0
129  REM  IF C=0 COMPUTER HAS LOS
     T
130 I = W
131 J = X
141  GOSUB 1000
145  REM  SCAN BOARD FOR BEST MOV
     E
150  REM  NOTE LIMITS TO SPEED UP
      PROGRAM
160 Q =  - 1
161  FOR I = N - 1 TO O + 1
162  FOR J = 5 TO 23
200  IF T(I,J) > 0 THEN 220
205 A = M(I - 4,J - 4)
210  IF A < Q THEN 220
215 W = I:X = J:Q = A
220  NEXT J: NEXT I
299  CALL  - 936: VTAB 21: HTAB 5
     : PRINT "MY MOVE = ";W - 4;"
     ,";X - 4
300 T(W,X) = 1:PI = W - 4:PJ = X -
     4:S = 1: GOSUB 800
301  IF M(W - 4,X - 4) < 100 THEN
      GOTO 97
307  CALL  - 936: VTAB 22: HTAB 5
     : PRINT "** I WIN **": GOTO
     35
310  IF C = O THEN  CALL  - 936: VTAB
     22: HTAB 5: PRINT "** YOU WI
     N **"
330  GOTO 35
799  REM  SUBROUTINE TO DISPLAY B
     OARD
800 SI = (PI - 1) * 8 + 64
802 SJ = (PJ - 1) * 8 + 4
804  DRAW S AT SI,SJ
806  FOR MM = 1 TO 4
808  FOR ZZ = 0 TO 100: NEXT ZZ
810  XDRAW S AT SI,SJ
```

```
812  NEXT MM
815  RETURN
990  REM  SUBROUTINE TO CALCULATE
      BEST MOVE
991  REM  SCAN THROU. MOVE AT I,J

992  REM  FOR FIVE SQUARES EITHER
      SIDE OF MOVE
993  REM  IN EIGHT DIRECTIONS,AND
      UPDATE BEST MOVE ARRAY
1000 K = 1:L =  - 1
1002  IF I < N THEN  IF I > 5 THEN
     N = I
1003  IF I > O THEN  IF I < 23 THEN
     O = I
1004  REM  UPDATE SCAN LIMITS
1005 U = I:V = J
1007  REM  I,J IS MOVE TO CHECK,
     D IS LOOP COUNT
1008  REM  K,L ARE X AND Y DIRECT
     IONS THRU MOVE
1010 D = 0
1011 D = D + 1
1013 P = 81
1020  REM  CHECK STILL ON BOARD
1026  IF U > 23 THEN 1090
1027  IF V > 23 THEN 1090
1028  IF U < 5 THEN 1090
1029  IF V < 5 THEN 1090
1030 E = U - 4:G = V - 4
1032 A = M(E,G):Q = T(U + K,V + L
     )
1034  REM  CALCULATE PRIORITYY OF
      POSITION
1035 R = T(U - K,V - L) * 27 + T(
     U - 2 * K,V - 2 * L) * 9
1036 R = R + T(U - 3 * K,V - 3 *
     L) * 3 + T(U - 4 * K,V - 4 *
     L)
1037 B = Q * 27 + T(U + 2 * K,V +
     2 * L) * 9 + T(U + 3 * K,V +
     3 * L) * 3 + T(U + 4 * K,V +
     4 * L)
1038  IF R = 80 THEN  IF T(U,V) =
     2 THEN C = 0
```

```
1039  IF T(U,V) <  > 0 THEN 1075
1040  REM  S(R) IS PRIORITY;THE F
     OLLOWING AER EXCEPTIONS
1042  IF R < 14 THEN  IF R > 11 THEN
      IF Q = 1 THEN P = 37
1044  IF R > 71 THEN  IF B > 53 THEN
      IF B < 63 THEN P = 80
1046  IF R > 71 THEN  IF B > 71 THEN
     P = 80
1048  IF R > 53 THEN  IF R < 63 THEN
      IF Q = 2 THEN P = 72
1050  IF P = 72 THEN  IF R = 60 THEN
     P = 31
1052  IF Q <  > 2 THEN  GOTO 1058

1053  IF R = 78 THEN P = 80
1054  IF R = 79 THEN P = 80
1056  IF R = 41 THEN R = 81
1058  IF R < 42 THEN  IF R > 35 THEN
      IF Q = 1 THEN P = 41
1059  IF R < 33 THEN  IF R > 29 THEN
      IF Q = 1 THEN P = 41
1060  IF R > 53 THEN  IF R < 63 THEN
      IF B > 71 THEN P = 80
1061  IF R > 38 THEN  IF R < 42 THEN
      IF Q = 1 THEN R = 40
1062  IF R > 35 THEN  IF R < 45 THEN
      IF B > 35 THEN  IF B < 45 THEN
     R = 40
1063  IF R > 27 THEN  IF R < 54 THEN
      IF B > 38 THEN  IF B < 42 THEN
     R = 40
1064  IF R = 79 THEN  IF A = 51 THEN
     M(E,G) = 41
1065  IF R = 0 THEN R = 81
1066  IF S(P) > S(R) THEN R = P
1067  IF S(R) -  INT (S(R) / 10) *
     10 = 1 THEN  IF A -  INT (A /
     10) * 10 = 1 THEN  IF S(R) <
     41 THEN R = 74
1068  IF S(R) -  INT (S(R) / 10) *
     10 = 9 THEN  IF A -  INT (A /
     10) * 10 = 9 THEN  IF S(R) <
     65 THEN R = 37
1069  REM  UPDATE BEST MOVE ARRAY
```

```
1070  IF S(R) > M(E,G) THEN M(E,G
    ) = S(R)
1075  IF D > 4 THEN 1090
1081 U = U + K:V = V + L
1085  GOTO 1011
1089  REM  CHANGE DIRECTION
1090  IF K = 0 THEN  IF L =  - 1 THEN
      RETURN
1095  IF K =  - 1 THEN  IF L =  -
    1 THEN K = 0
1100  IF K =  - 1 THEN  IF L = 0 THEN
    L =  - 1
1105  IF K =  - 1 THEN  IF L = 1 THEN
    L = 0
1110  IF K = 0 THEN  IF L = 1 THEN
    K =  - 1
1115  IF K = 1 THEN  IF L = 1 THEN
    K = 0
1120  IF K = 1 THEN  IF L = 0 THEN
    L = 1
1125  IF K = 1 THEN  IF L =  - 1 THEN
    L = 0
1130  GOTO 1005

]FP
```

附　　　錄

附錄一　INTEGER BASIC 和 APPLESOFT 之比較
附錄二　參考書目

附錄一 INTEGER BASIC和 APPLE SOFT之比較

每一位擁有 APPLE Ⅱ的朋友大概都會APPLESOFT BASIC，至於INTEGER BASIC，可能大家都〃聽過〃，但詳細的情形便不太了解了。且往往我們在書中或雜誌上看到一篇很好的程式〃只可惜是用INTEGER BASIC寫的〃。附錄一就是針對這些問題，藉著與 APPLESOFT BASIC比較的方式，希望讀者能對INTEGER BASIC不僅僅〃聽過〃而已，更能將自己在書上或雜誌上所看到的INTEGER BASIC程式轉換成APPLESOFT BASIC(若讀者沒有INTEGER CARD)。當然若讀者對INTEGER BASIC很熟悉則此附錄便可跳過或僅供參考之用。

附錄一假設讀者對INTEGER BASIC不太熟悉，但對於APPLESOFT BASIC則有某些程度的了解。首先我們由這兩種語言對變數名稱及 DIMENSION 指令的不同處理方式開始向下討論：

(1)變數名稱（VARIABLE NAME）：

兩種語言均規定變數名稱不得與保留字（Reserved Word）重覆，且變數名稱的第一個字元必須是A－Z的字元。而其主要差別則在於APPLESOFT只辨認變數名稱的頭二個字元，而INTEGER BASIC則辨認整個變數名稱。如INTEGER認為APPLESAUCE＄與APPLEJUICE＄分別代表不同的變數名稱

，但APPLESOFT則認爲這二個字串變數爲同一字串變數：AP$。

(2)DIMension指令：

兩種BASIC 語言對與字串的處理極不相同。在 INTEGER 中程式設計者在使用一個字串變數之前，必須用DIM指令宣告該字串之最大長度；且INTEGR不允許字串陣列，故利用DIM宣告字串變數與整數變數所代表的意義並不相同。

而在字串的處理方面，INTEGE 沒有 STR $或 VAL 指令，故無法對字串與數值資料之間做轉換。但兩種語言對 LEN 指令的處理方式相同（這大概是兩者在字串函數方面唯一的相同點）。A$=A$+B$在APPLESOFT中代表的意義爲：將字串A$與字串 B$頭尾相接後重新定爲字串 A$；但在 INTEGER 中則必須改成A$（LEN（A$）+1）=B$。

其次我們考慮兩個指令：

PRINT　A$（7，14）　與　PRINT　A$（4）在APPLESOFT中會分別將字串陣列 A$中註標爲（7，14）及（4）的元素顯示出來。而在 INTEGER 中，由於沒有字串陣列，會分別將 A$字串中第 7 個至14個共 8 個字元顯示出來；而 PRINT A$（4）由於沒有指出結束位置，所以 INTEGER 將字串 A$中由第 4 個至最後一個字元全部顯示出來。且在 INTEGER中沒有MID$，RIGHT$或LEFT$等函數指令，而必須以上述方式達到這些類似的字串處理功能，如若我們想顯示出字串A$ 的最後四個字元在INTEGER中便須PRINT　A$（LEN（A$）-3，4）。

下面的程式一爲 INTEGER BASIC寫成，用以說明我們上面所討論過的一些情況；程式二則爲其所對應的APPLESOFT

BASIC 程式。讀者可在自己的APPLE Ⅱ 上試試看，還蠻有趣的！

程式一：

```
>LIST
10   DIM TEST1$(15), TEST2$(15), T$(30)
20   PRINT "ENTER FIRST TEST STRING" :
     PRINT
25   INPUT TEST1$ : PRINT
30   PRINT"ENTER SECOND TEST STRING" :
     PRINT
35   INPUT TEST2$ : PRINT
40   FOR X = LEN ( TEST1$ ) TO 1 STEP-1 :
     Y = Y + 1 : T$ ( Y ) =TEST1$ ( X, X ) :
     NEXT X
50   PRINT T$ : REM REVERSE STRING
70   T$=TEST1$ ( 1 , 4 )
80   T$ ( 5 )=TEST2$ ( LEN ( TEST2$ ) - 3 )
90   PRINT T$ : REM JOIN FIRST 4 CHAR·
     OF TEST1$ AND LAST 4 CHAR·OF
     TEST2$
100  END
```

程式二：

```
]LIST
10   REM
20   PRINT "ENTER FIRST TEST STRING" :
     PRINT
```

```
25   INPUT T1$ : PRINT
30   PRINT "ENTER SECOND TEST STRING" :
     PRINT
35   INPUT T2$ : PRINT
40   FOR X=LEN(T1$) TO 1 STEP-1 :
     T$=T$+MID$(T1$,X,1):NEXT X
50   PRINT T$ : REM REVERSE STRING
70   REM
80   T$=LEFT$(T1$,4)+RIGHT$(T2$,4)
90   PRINT T$ : REM JOINT FIRST 4 CHAR·
     OF T1$ AND LAST 4 CHAR·OF T2$
100  END
```

(3)分支指令（Branch Statements）：ON…GOTO…

在 INTEGER中任一敍述中不可使用多重分支指令（multiple branch），僅能使用如GO TO 100或IF-THEN GO TO 100等。但可以使用計值 GO-TO如 IF A<4 THEN GO TO A*10或GO TO N，其中A* 10 即爲GO TO 的行數，利用這種方式便可達到多重分支的目的。而在APPLESOFT中不可使用計值 GO TO，任何分支指令之後必須跟著整數的行數且不可大於 63999。因此若我們在轉換 INTEGER BASIC程式時，"不幸"碰到這類計值 GO TO時，首先應仔細判斷此計值 GO TO可能 GO TO到那些行數去？控制的變數值爲何？再利用APPLESOFT所具有的多重分支指令（如ON A GO TO 100，200，300）來達成這個目的。

此外必須注意一點若APPLESOFT的 ON exp GO TO 所計算之exp值小於 1 或大於 GO TO 後面所列出的行數個數，則此

GO TO 指令不被執行。又APPLE SOFT 中的 ON-GOSUB 情況與ON-GOTO類似。

(4)一行上擁有一個以上的敍述句(Multi-Statement Lines)

兩種語言均允許一行上擁有數個敍述句，且處理方式十分相似，僅有一種情形例外。而作者認爲這種情況最令人頭痛，往往使得我們在轉換這兩種語言所寫的程式時，必須對程式做大幅度的修改。

在 INTEGER中，當一個 IF-THEN 敍述句(statement)被判斷爲 FALSE 時，程式會跳過此敍述句，進行下一個敍述句(不論此敍述句與 IF-THEN 敍述句是否在同一行上)。但在APPLESOFT中當 IF-THEN 敍述句被判斷爲 FALSE 時，程式會跳過此 IF-THEN 敍述句所在之行，進行下一行(NEXT LINE NUMBER)。

APPLESOFT所採用的這種方式，依作者個人的使用經驗覺得較 INTEGER的方式好。這種方式可避免一些分支的情況，增加執行速度，且使得整個程式較具有結構。

總之在轉換程式時，若碰到此類 IF-THEN 敍述句，我們必須特別小心，注意程式的邏輯才不致發生錯誤。

(5)游標的移動

在游標的移動方面，兩種語言的處理方式差不多，但仍有些地方應注意：

①CALL-936

無論在 INTEGER 或 APPLESOFT，CALL-936均使文字幕被清除，游標移到螢幕的左上角。但APPLESOFT可用HOME指令代替，而 INTEGER沒有此指令。

②TAB 20

在INTEGER中，TAB 20將游標移到距螢幕左端20個位置的地方。若要達到相同的目的，在APPLESOFT中應使用HTAB 20。在APPLESOFT中雖然也有TAB指令但必須被用在PRINT敍述句中且數值必須用小括號括起來（TAB（20））如：

INTEGER STATEMENT……

10 TAB 12：PRINT "APPLE"

APPLESOFT equivalents……

10 HTAB 12：PRINT "APPLE"

OR

10 PRINT TAB（12）"APPLE"

(6)其他一些用法上的差別：

①INTEGER： 100 INPUT "ENTER DATA"，A

APPLESOFT： 100 INPUT "ENTER DATA"；A

兩種型式差不多，但APPLESOFT必須用分號（；）代替INTEGER的逗點（，）。

②INTEGER： 100 IF A#B THEN GOTO 200

APPLESOFT： 100 IF A<>B THEN GOTO 200

對於"不等於"的符號APPLESOFT必須用"<>"（大於且小於代替INTEGER的"#"的符號。

③INTEGER中的POKE 50,63 相當於APPLESOFT的INVERSE

INTEGER中的POKE 50,127相當於APPLESOFT的FLASH

INTEGER中的POKE 50,255相當於APPLESOFT的NORMAL

④MOD FUNCTION

APPLESOFT中沒有MOD內儲函數（BUIDIN-FUNCTICN），因此對於INTEGER 中的MOD函數我們可用下列方式轉換：

IN INTEGER…… A MOD B

APPLESOFT…… MOD（A）= INT（（A／B－INT（A／B）* B + .05）* SGN（A／B）

⑤清除變數內容：

如一INTEGER敍述句：

10 CLR：FOR X＝1 TO 35： A（X）＝0：NEXT

這類型的敍述句有時候我們可在INTEGER程式的開頭或遊戲程式中看到。CLR將所有的數值變數（Numeric Var.）設定為 0 並將所有字串變數設定為空字串，但對於陣列變數却沒有影響，而必須用後面的FOR……NEXT廻圈清除。在APPLESOFT中我們可用 10 CLEAR 取代，不但能將所有變數，且能將陣列變數一起清除。

⑥SCREEN PRINTING

讓我們考慮下面的INTEGER的程式行：

10 FOR X=1 TO 5 ： PRINT "*",：NEXT X

及APPLESOFT的：

10 FOR X=1 TO 5 ： PRINT "*",：NEXT X

兩者完全相同，但它們的輸出結果却不相同。這類型指令通常用來將資料顯示在螢幕上的一些特定位置，可節省不少 TAB-PRINT 類型的敍述句。在INTEGER中此“*”字元會顯示在螢幕上同一列的 1，9，17，25，33各行上；而

在APPLESOFT中此“ * ”字元則會被分別顯示在螢幕上的二列上，前三個“ * ”顯示在第一列的 1 ，17 ，33 行上，後二個“ * ”則被顯示在第二列的 1 和17行上。因此若我們在轉換 INTEGER 程式爲APPLESOFT程式時，應注意對螢幕上的輸出格式重新安排。

(7)記憶體的使用

最後我們所將討論的差別與語法沒有什麼關係，爲記憶體的使用方式，若在程式中沒有使用到圖形表或呼叫一些使用者自定的機器語言程式，則大可不必理會此記憶體的使用方式，可直接轉換兩種程式；但若有使用到，則我們必須注意是否必要將其移動位置或改變指標。

以下討論爲 48K APPLE Ⅱ PLUS 系統且有 APPLE DOS占用部份記憶位置的情形（配合第二章所討論的 APPLESOFT 記憶位址的配置）。

在 INTEGER 中，程式由 HIMEM：38400 向下（低位址）儲存；若一旦轉變爲APPLESOFT，程式則由LOMEM：2048 向上（高位址）儲存。因 INTEGER中沒有 HIMEM：及 LOMEM：指令，故 INTEGER 程式若使用到機器語言程式或圖形表，可在程式中利用 POKE LOMEM：的方式將其置於位址 2048 和 LOMEM之間，或在記憶體中找尋其他位址存放。如在海空大戰的 INTEGER程式中加入： 0 POKE 204，0：POKE 205，16：POKE 74，0：POKE 75，16 設定LOMEM：4096 留下足夠的位置給機器語言程式或圖形表使用。

在將此類程式轉變爲APPLESOFT程式時，便應特別注意這點差別，否則可能會使部份APPLESOFT程式與機器語言程式或圖形表重疊。通常我們可在APPLESOFT 中將HIMEM：向上

移，留下HIMEM：以上的記憶區給這些圖形表或機器語言程式占用，或不移動這些圖形表及機器語言程式，而將APPLESOFT程式由 4096 向上儲存。

以上我們討論了許多APPLESOFT與 INTEGER BASIC的差別，還有如 INTEGER中的CHAIN指令沒有討論，因作者覺得這些指令通常不會遇到，故不再討論，有興趣的讀者可自行參閱有關書籍。寫到這裡相信讀者對 INTEGER BASIC的使用方式都能有一些概念了。利用這種方式大概可稱得上所謂“INTEGER BASIC速成”了吧！

附錄二　參考書目

Ref 1：APPLE COMPUTER INC, APPLE Ⅱ REFERENCE MANUAL

Ref 2：APPLE COMPUTER INC, BASIC PROGRAMMING REFERENCE MANUAL APPLESOFT

Ref 3：PEEKING AT CALL-A.P.P.L.E. VOL 1-2

Ref 4：PEEKING AT CALL-A.P.P.L.E. VOL 3

Ref 5：CALL-A.P.P.L.E. In Depth , Number one

Ref 6：Jeffrey Stanton , APPLE GRAPHICS & ARCADE GAME DESIGN

Ref 7：NIBBLE EXPRESS , VOL. 1. 11

Ref 8：William E. Dougherty , THE APPLE Ⅱ MONITER PEELED

Ref 9：APPLE Ⅱ THE DOS MANUAL：DISK OPERATING SYSTEM

Ref10：APPLE Ⅱ 圖形遊戲（GRAPHIC GAME）系統設計　林和賢編著，波前電腦圖書出版

Ref11：APPLE Ⅱ 微電腦徹底研究（五）動畫技巧篇，施威銘著，波前電腦圖書出版。

著作權、版權所有・翻印必究

法律顧問：邱晃泉律師

從APPLEⅡ到圖形遊戲製作

編著者：宗世麟
發行人：劉麗雪
出版者：波前電腦管理圖書有限公司
地　址：台北市泰順街2巷30號
電　話：3927101～2
郵　撥：0561411-5 帳戶：劉麗雪
定　價：台幣180元
印刷者：名揚印刷有限公司
中華民國　73年　9月　出版

局號：　　收據號碼：

郵政劃撥儲金存款單

收款帳號	收款戶名	新臺幣	寄款人姓名住址
0561411-5	劉麗雪 台北市泰順街2巷30號	（請用大寫數目字填寫並於數末加一整字）	
經辦局郵戳			中心局郵戳
			手續費　次　元

主管：

經辦：

本聯由劃撥儲金組存查

請注意：一、帳號、戶名及寄款人姓名住址請詳細填明，以免誤寄。
二、抵付交換票據之存款，務請於交換前一、二天存入，必要時，可請存款局先以電話通知劃撥儲金組，惟長途電話費由存款人負擔。

郵政劃撥儲金存款通知單

收款帳號	收款戶名	寄款人姓名住址	新臺幣
0561411-5	劉麗雪 台北市泰順街2巷30號	姓名： 地址： 電話：	（請用大寫數目字填寫並於數末加一整字）
經辦局郵戳		手續費　次　元	

主管：

經辦：

本聯經劃撥儲金組登帳後寄交帳戶

存款後由郵局掣給正式收據為憑，本單不作收據用

波前電腦提供您磁片、卡帶服務

■APPLE Ⅱ徹底研究㈤磁片　動畫技巧篇（特價250元）

- 專業的技法，動畫的極至，附使用手册。
- 包括全書所有程式（改良版），附LISA組譯器。
- 使用本磁片，便可以寫出自己的 GAME 來。

■從APPLE Ⅱ到圖形遊戲製作（磁片）（特價 180元）

- 包含太空迷陣、金字塔、……共十個GAME。
- 包含全書主要公用程式。
- 省却Key－In 的麻煩。

■APPLE Ⅱ圖形遊戲系統設計（磁片）（特價250元）

- 包含大蜘蛛、金樸克、……等十個GAME。
- 磁帶系已錄製成自動讀造形表形式。
- 節省您Key－In 的時間，更好的學習效果。

此欄係備寄款人與帳戶通訊之用，惟所作附言應以關於該次劃撥事項爲限。

通信欄

茲向貴公司購買

□APPLE Ⅱ徹底研究㈤磁片

□從APPLE Ⅱ到圖形遊戲製作磁片

□APPLE Ⅱ圖形遊戲系統設計磁片

寄款人：

封面設計／施威銘
美工完稿／嚴　珣

介紹您：波前的新書

學習電腦的一大步

Apple II 組合語言速成　章 豪 譯
Apple II 組合語言精華　李天佑著

高手的挑戰

Apple II 磁片 COPY 大全
（附實例技巧解析）

本書的姐妹作

Apple II 圖形遊戲系統設計　林和賢著

動畫GAME的極至

Apple II 徹底研究(五)　施威銘著
（動畫技巧篇）

・波前出版・必屬好書・

這本書將……

] ■告訴您有關APPLE SOFT的製作方法。

■告訴您如何輕易得使用APPLE發出優美動聽的音樂。

■告訴您如何輕易得製作高、低解像圖形。

■告訴您如何寫出一個大型結構化的程式。

■用深入淺出的方式，引導您進入圖形遊戲的領域。

波前電腦管理圖書有限公司

MCA-010

從APPLE II到圖形遊戲製作(GRAPHIC GAME)

宗世麟 著

波前